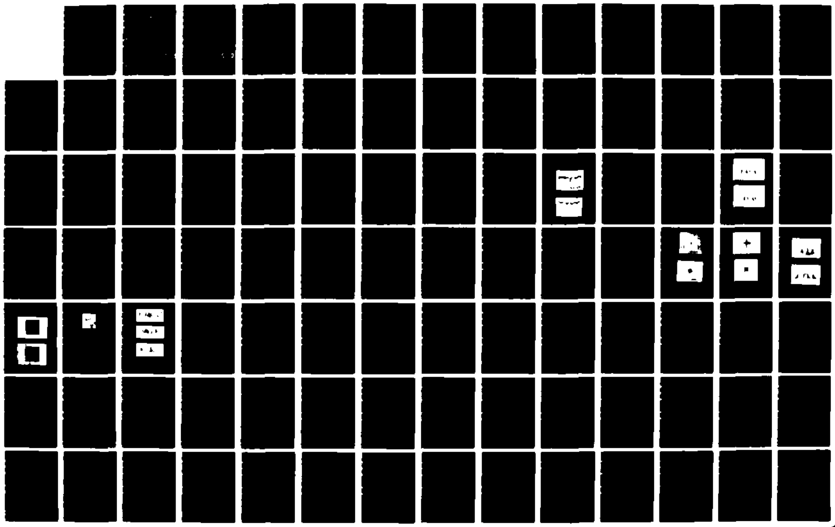
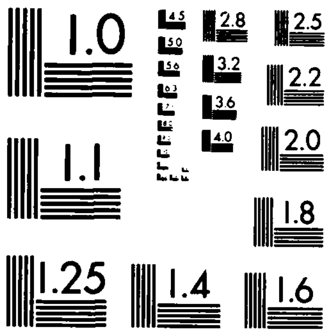
AD-A189 553 COMPUTER GENERATED HOLOGRAM AND MAGNETO-OPTIC SPATIAL LIGHT MODULATOR FOR (U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI M W MAYO 1/2

UNCLASSIFIED DEC 87 AFIT/GEO/ENG/87D-1 F/G 12/9 NL




1.0
4.5
5.0
5.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

AF INSTITUTE OF TECHNOLOGY

# Computer Generated Hologram and Magneto-optic Spatial Light Modulator for Optical Pattern Recognition

THESIS

Michael W. Mayo
Second Lieutenant, USAF

AFIT/GEO/ENG/87D-1

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GEO/ENG/87D-1

# Computer Generated Hologram and Magneto-optic Spatial Light Modulator for Optical Pattern Recognition

THESIS

Michael W. Mayo
Second Lieutenant, USAF

AFIT/GEO/ENG/87D-1

AFIT/GEO/ENG/87D-1

# Computer Generated Hologram and Magneto-Optic Spatial Light Modulator for Optical Pattern Recognition

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Michael W. Mayo

Second Lieutenant, USAF

December 1987

## Preface

The purpose of this study was to develop and test an optical position, scale, and rotation invariant pattern recognition system using a computer generated hologram and spatial light modulator. This research contributes to the Air Force's on-going investigation of autonomous target recognition systems.

I received a tremendous amount of help and encouragement while working on this thesis. First, I am deeply indebted to my thesis advisor, Dr. Matthew Kabrisky, for his patience and understanding help with all aspects of the research. I would also like to thank the members of my committee, Dr. Steven K. Rogers and Dr. James P. Mills, for their guidance and timely suggestions, especially with the optics. My gratitude goes to Captain Robert Williams for his insightful comments and extended help with the digital design and modifications of the Litton spatial light modulator. I would also like to thank Captain Dave Miazza for his guidance and help with the overall project.

Finally, and most importantly, I would like to thank my wife, Carey, for her understanding, encouragement, and help during this thesis effort.

"Original contains color plates: All DTIC reproductions will be in black and white"

## Table of Contents

## List of Figures

## List of Tables

## Abstract

This thesis investigates the integration of an optical system for real-time position, scale, and rotation invariant pattern recognition. Specifically a Litton Magneto-Optic Spatial Light Modulator is interfaced to a Zenith 248 microcomputer and AT&T frame grabber. A user interface written in C allowed arbitrary patterns to be written to the SLM by a user. Patterns can be generated in computer code, transferred from a CCD camera (via the frame grabber), taken from an image file on computer disk, or taken from a VAX image file. The ability to locally generate a computer generated hologram using the computer generated interferogram method on the Sun workstations and Imagin laser printer was developed. The CGH developed was used in an optical system which performed a Fourier transform and lnr-$\theta$ coordinate transformation to create a position, scale, and rotation invariant feature space. CCD cameras were used throughout the optical processor for display/analysis. The entire system was developed and tested using various templates and real scenes.

# COMPUTER GENERATED HOLOGRAM AND SPATIAL LIGHT MODULATOR FOR OPTICAL PATTERN RECOGNITION

## I. Introduction

Automatic, real-time pattern recognition is a major concern of the Air Force in the area of pilot aids, smart weapons, and general purpose robotics. A pattern recognition system that works in real-time and operates independently of the position, scale, and in-plane rotation of the input target is needed to accurately find and identify targets embedded in their natural surroundings. Pattern recognition to date has been implemented primarily on large computer systems employing algorithms which require an enormous amount of calculations, leading to a time-consuming task. Pattern recognition, performed by a coherent optical system, allows for parallel processing and high speed calculations (nanoseconds). These advantages over electronic serial processing techniques make the use of optics appealing for the pattern recognition task.

This thesis documents an effort to implement portions of a position, scale, and rotation invariant, real-time, optical pattern recognition system using computer-generated holograms (CGH) and a magneto-optic spatial light modulator

(SLM). The SLM will be used to input real-time scenes into the system, while the CGH, along with a Fourier transform (FT) lens, will perform the lnr-$\theta$ coordinate transformation. The general procedure to be followed in solving the optical pattern recognition problem will be discussed in the Approach section of this thesis.

## Summary of Current Knowledge

Many pattern recognition systems (both digital electronic and optical) rely on correlation using a template matching scheme to "recognize" a given input. Correlators have the inherent ability to accomodate translational shifts, and to simultaneously handle multiple targets. A high intensity peak is produced during autocorrelations (correlation with identical target) whereas low intensity distributions arise during cross-correlations (correlation with different targets). However, correlators are very sensitive to changes or distortions between the ideal reference and the actual input scene.

The optical correlation of two 2-dimensional images is achieved in parallel and real-time by placing the conjugate Fourier transforms of reference objects into the transform plane of an optical processor like the one shown in Figure 1.1. In this system, the Fourier transform of the input scene is multiplied by the conjugate FT of the reference object. The correlation of the two space functions appears in the output plane (2:027). The use of spatial light modulators (SLM) and liquid crystal televisions (LCTV) allows a system to handle real-time input scenes while performing pattern recognition with multiple reference objects or targets (8:466, 12:698-704, 11:467-468).

The conventional VanderLugt-type correlator used for pattern recognition is unable to recognize scaled or rotated images of a reference object. It has been shown that a 1% scale change of the reference object causes the signal to noise ratio of the resultant correlation peak to decrease by 10dB from that of the autocorrelation. Likewise, a 20dB loss occurs for a 2° rotation of the input from the reference (2:1652). One solution is to perform a lnr-$\theta$ coordinate transformation preprocessing operation, which results in partial scale and rotation invariance (5:289).

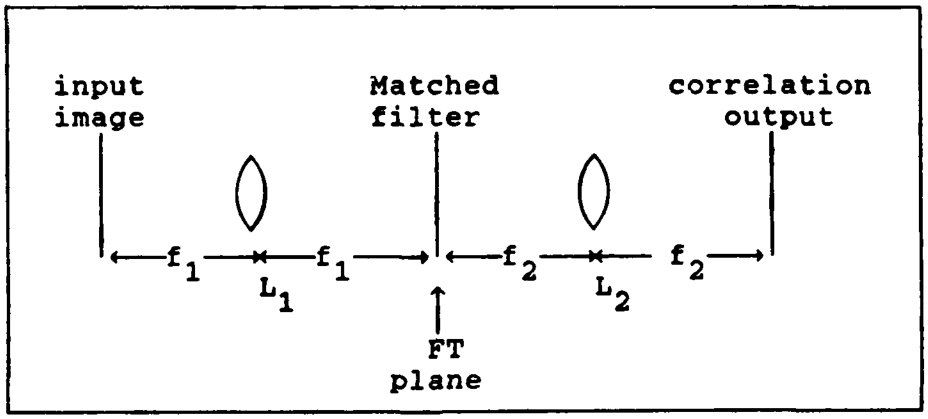


Figure 1.1. VanderLugt correlator

A low cost, commercially availible LCTV has been used as a SLM in a VanderLugt-type coherent optical correlator (Figure 1.1). Results have shown that the LCTV resolution, contrast, and speed are sufficient for basic (geometrical objects in low noise fields) real-time pattern recognition applications (11:467). The LCTV's main advantage is its versatility. It can be used in the input plane and/or in the transform plane, and it can be addressed from a remote video camera, a personal computer, or an external transmitter via a standard antenna. Disadvantages of using the LCTV in high resolution processes include: (1) the inability to obtain more than a 10dB dynamic range (12:41), (2) the introduction of phase distortion in the passing wavefronts (6:938), and (3) its sensitivity to the polarization angle of the incident light (12:72).

The Litton iron garnet, H-triggered, magneto-optic device (LIGHT-MOD™) has been used in similar capacities in more expensive optical signal processing systems where greater dynamic range, resolution, and speed are necessary (8:466, 12:698-704, 16:55-64). The LIGHT MOD functions predominantly as a spatial light modulator (SLM). The data (either scene or filter) are transferred from a personal computer to the device through the horizontal and vertical electrode structure of the LIGHT MOD. Since each pixel or line can be switched in approximately 1$\mu$sec, a 512x512 frame can be transferred to the device in 0.512 msec in parallel (17:58). By using the parallel addressing scheme, scenes can be input quickly to the processor when the LIGHT MOD is positioned at the input plane. Psaltis used two SLM's in a

VanderLugt correlating configuration, which illustrates the usefulness of the LIGHT MOD functioning as a SLM. The input image was recorded on a SLM placed in the image plane of Figure 1.1, whereas the FT of a reference image (acting as a matched filter) was recorded on the second SLM placed in the transform plane (16:699).

The recent use of computer generated holograms (CGH's) in optical processors has allowed systems to perform coordinate transformations in nanoseconds as opposed to the required calculations taking tens and hundreds of seconds on electronic digital computers (3:217-222, 6:938-942, 17:8). Computer generated phase hologram masks with a given phase transfer function can be used to produce various geometrical transformations. The desired coordinate transformation for this thesis was the $\ln r-\theta$.

## Problem Definition

The thesis problem is the development of portions of an optical pattern recognition system which can locate a given target independent of its position, scale, or rotation in realistic input scenes. The system must be able to recognize and locate the target in real-time. The real-time goal requires integration of a CGH and a computer controlled SLM.

## Scope of Thesis

This thesis focused on constructing a real-time, position, scale, and rotation invariant (PSRI) optical pattern recognition system to identify arbitrary targets in low noise scenes. The integration of a CGH intc an optical system and the control of a SLM by a computer was a main thrust in constructing the real-time system. In general, a generic optical processing lab was established to carry-out the ongoing pattern recognition effort at AFIT.

## Approach

The thesis problem was split into three different parts. The first part included the LIGHT MOD modifications, software development for the LIGHT MOD, and an investigation of using the LIGHT MOD as an input and transform plane device. Particular emphasis was placed on integrating the LIGHT MOD into the Zenith 248/AT&T frame grabber/Sony CCD camera system and creating a user friendly environment.

Part two concentrated on reconstructing a rotation and scale (but not position) invariant optical system as documented in the literature (4; 5; 13). The development and reduction of a lnr-$\theta$ CT CGH was included. Results were compared to those obtained by others in the literature to verify the validity of the CGH.

Part three was the PSRI optical system implementation. Targets were placed into the system as transparencies or cutouts. The magnitude squared of the Fourier transform of the input image was detected by a CCD camera and sent to the LIGHT MOD which was functioning as a SLM. The magnitude of the Fourier transform was then used as the input to a

$\ln r$-$\theta$ coordinate transformation computer generated hologram in order to construct the position, scale and rotation invariant feature space. This feature space could then be used as the input to an optical correlator. An optical correlator was constructed during the thesis effort but no correlating results were obtained due to insufficient time.

# II. Operation of the SLM

In this chapter, properties and modifications of the Litton iron-garnet, H-triggered magneto-optic device (LIGHT-MOD) are characterized. In the first section, the construction and operation principles of the LIGHT MOD will be discussed. The second section addresses the modifications required to use the LIGHT MOD with the Zenith 248 computer system. The last section briefly discusses the performance of the LIGHT MOD.

The LIGHT MOD can be used in optical signal processing as an electrically alterable high speed two-dimensional spatial light modulator (SLM). Such a SLM can function as an electrical-optical interface to input images into the object plane of an optical system, to place spatial filters into the Fourier transform plane, and to modulate or scan the processed output image or wavefront. The LIGHT MOD in this thesis effort functioned as a SLM to input real-time scenes as well as to strip the phase from the Fourier transform of input scenes. The magnitude squared of the Fourier transform was displayed on the SLM and used as the input to the lnr-$\theta$ coordinate transform CGH (CT CGH).

## Construction and Basic Operation Principles

The LIGHT MOD consists of a bismuth substituted transparent iron garnet film grown on a nonmagnetic substrate. The direction of uniaxial anisotropy is oriented perpendicular to the plane of the film. Pixels are set in even rows and columns (128 rows by 128 columns - 16,384 total pixels) and metallic gold conductors are deposited along one edge covering the full length of the rows and columns. The conductors cross at one corner of each pixel where a small loop in the conductor increases the magnetic flux due to current flow in the conductor (17: 55-56).

Operation of the LIGHT MOD when used as an SLM or light valve is depicted in Figure 2-1. A source of polarized light is required as an input and an analyzer is placed on the opposite side of the light source. As the polarized light passes through the pixels, the polarization of the transmitted light is rotated clockwise or counterclockwise depending upon the magnetization within the pixel. The amount of the resulting rotation depends upon the Faraday constant of the material and the thickness of the film (17:58). The analyzer is rotated with respect to the polarization of the input (laser source) so that light will

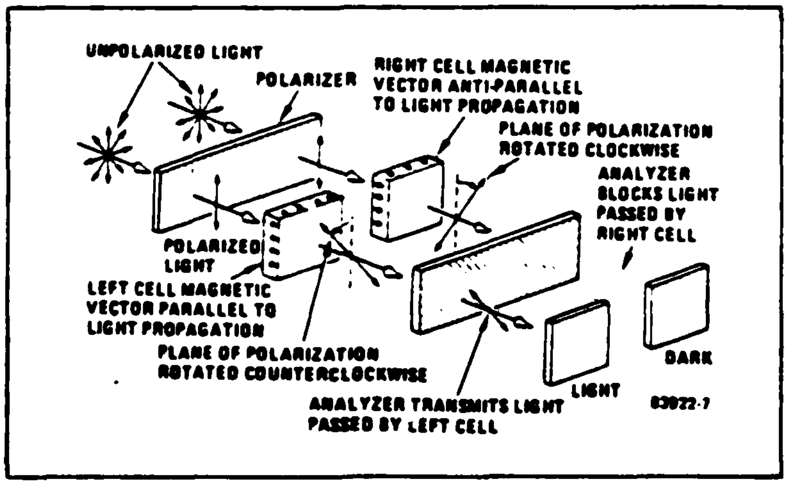

UNPOLARIZED LIGHT
POLARIZER
RIGHT CELL MAGNETIC VECTOR ANTI-PARALLEL TO LIGHT PROPAGATION
PLANE OF POLARIZATION ROTATED CLOCKWISE
ANALYZER BLOCKS LIGHT PASSED BY RIGHT CELL
POLARIZED LIGHT
LEFT CELL MAGNETIC VECTOR PARALLEL TO LIGHT PROPAGATION
PLANE OF POLARIZATION ROTATED COUNTERCLOCKWISE
DARK
LIGHT
ANALYZER TRANSMITS LIGHT PASSED BY LEFT CELL
83022-7

Figure 2.1. Operation of LIGHT MOD as SLM (17: 57)

either pass through or be blocked, depending on the magnetic polarity of any given pixel. The analyzer is adjusted to set image polarity (light scene on dark background or vice versa) and contrast ratio (dynamic range).

A pixel is changed by simultaneously applying a current pulse to a selected x and y drive line. This current pulse nucleates or partial changes the selected pixel into a "neutral" state. This state must be followed by the application of an external magnetic field (coil pulse) to complete the change of the pixel to the on or off state. The application of the coil pulse is referred to as pixel saturation. The switched pixel will then remain in that state indefinitely until it is nucleated in the opposite direction by drive line currents.

## Modification of the SLM

Although the LIGHT MOD was to be a fairly small part of the optical pre-processing/pattern recognition system, the modifications required to get the LIGHT MOD operational and user friendly became a very large part of this thesis effort. The L135, 128x128 display LIGHT MOD purchased from Litton Data Systems is a developmental model designed for the Apple computer; however, the AFIT labs are equipped with Zenith 248 computers which are IBM AT compatible. Available to us was the interface board from Semetex Corporation. They make a similar SLM device which is driven by the IBM class computers; however, the Semetex interface board and the LIGHT MOD were not compatible. Instead of designing an interface board to link the Zenith 248 and LIGHT MOD, the existing outputs from the Semetex board were modified and used as the inputs to the LIGHT MOD driver cards.

A block diagram of the 128x128 LIGHT MOD display system is shown in Figure 2.2. Cable C1 connects the display

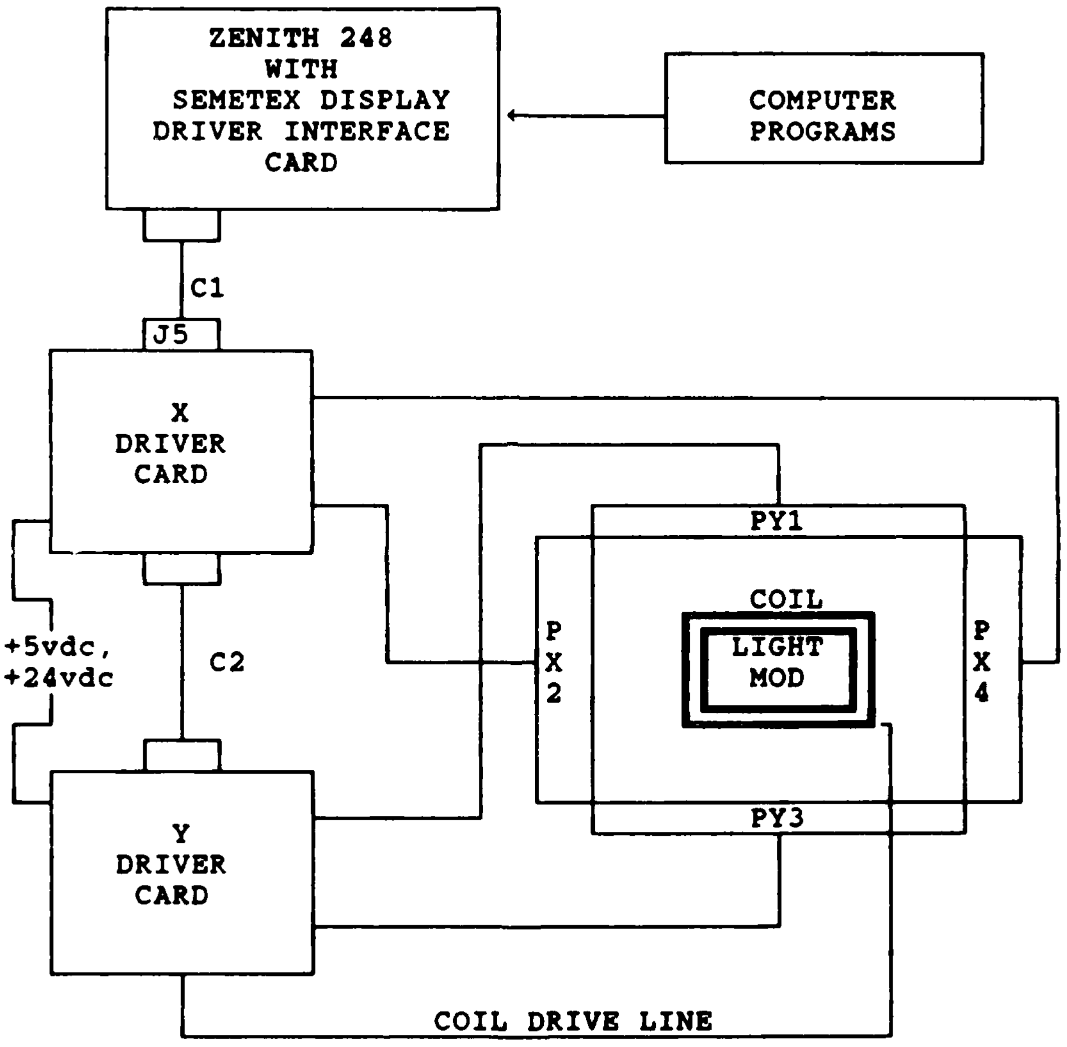


Figure 2.2. Block Diagram of 128x128 LIGHT MOD

driver interface card in the Zenith 248 to pin J5 on the X driver card. The necessary signals consist of X and Y addresses, write/erase control (SPA7), pixel nucleation pulse (write pulse, PT), pixel saturation pulse (coil pulse, PC), and grounds. Table 2.1 shows the required signals for the X driver card and their respective locations on pin J5 (a 26 pin connector).

Table 2.1
Required signals from computer interface to pin J5 on X Driver Card

| J5 pin number | Signal name | Cable signal function | |
|---|---|---|---|
| 1 | FPB0 | X Address | LSB |
| 2 | FPB1 | " | |
| 3 | FPB2 | " | |
| 4 | FPB3 | " | |
| 5 | FPB4 | " | |
| 6 | FPB5 | " | |
| 7 | FPB6 | " | MSB |
| 8 | FPB7 | Not Used | |
| 9 | FPA0 | Y Address | LSB |
| 10 | FPA1 | " | |
| 11 | FPA2 | " | |
| 12 | FPA3 | " | |
| 13 | FPA4 | " | |
| 14 | FPA5 | " | |
| 15 | FPA6 | " | MSB |
| 16 | FPA7 | Not Used | |
| 17 | SPA7 | Write\Erase Control | |
| 18 | SPARE | Not Used | |
| 19 | GROUND | Ground | |
| 20 | GROUND | Ground | |
| 21 | PT | XY Pulse | |
| 22 | PC | Coil Pulse | |
| 23 | SPARE | Not Used | |
| 24 | SPARE | Not Used | |
| 25 | SPARE | Not Used | |
| 26 | SPARE | Not Used | |

- MSB : most significant bit.
- LSB : least significant bit.
- *note: J5 connector is numbered with odd pins on the outside and even pins on the inside.

The seven X addresses (FPB0-6) and seven Y addresses (FPA0-6) correspond to $2^7 = 128$ rows and columns, respectively. For example, FPB0-6 = 000 0001 (binary) corresponds to row 1; FPA0-6 = 111 1111 (binary) corresponds to column 127 (rows and columns are numbered 0-127).

<u>Litton signals</u>

A pixel is written when the X and Y address lines are asserted and a positive logic level is applied to the write/erase control line (SPA7). SPA7 latched high (positive logic level - binary one) causes the coil current pulse (PC) to go in the clockwise direction, which turns the nucleated pixel on. If, on the other hand, SPA7 is latched low, the coil current pulses in the counterclockwise direction causing the nucleated pixel to turn off.

The timing of the above mentioned signals is extremely critical, as shown in Figure 2.3. TPT and TPC are the outputs of an SN74LS123 retriggerable monostable multivibrator located on the X driver card and are triggered by the positive edge of inputs PT and PC, respectively. A variable resistor is connected between $R_{int}$ and $V_{cc}$ on the SN74LS123 to obtain variable pulse widths for TPT and TPC. The pulse widths were set as shown in Figure 2.3 to obtain optimum results. (Excessive width of TPT will cause LIGHT MOD lines to fuse open - the pulse is typically 1$\mu$sec wide).

TPT must be pulsed while the X and Y addresses and SPA7 are valid in order to nucleate a pixel. TPC is then pulsed to saturate the given pixel.

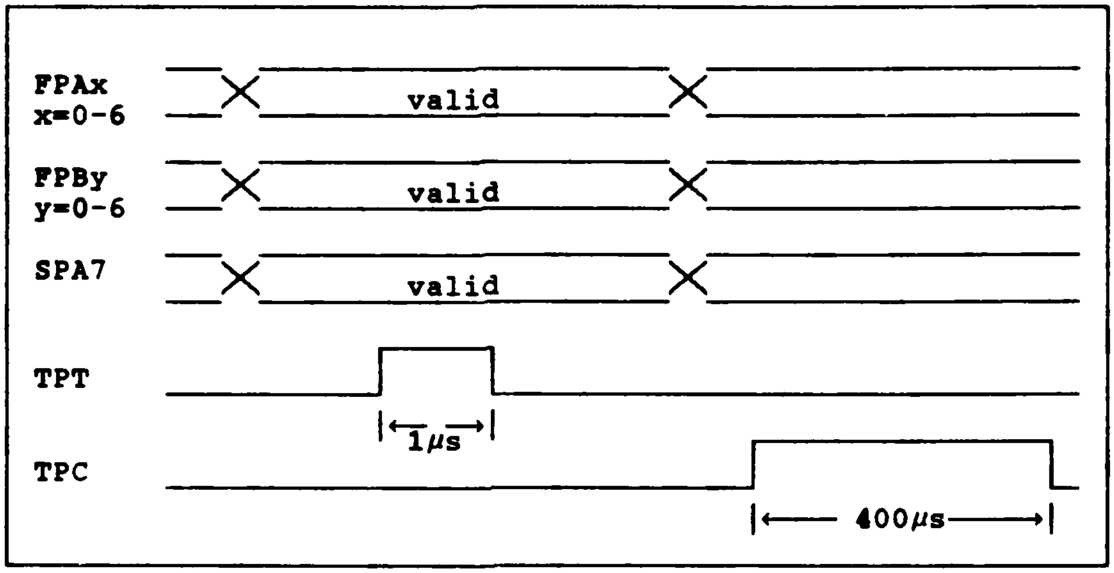


Figure 2.3. LIGHT MOD timing diagram (9).

<u>Interface with the Semetex board</u>

All the required signals could be obtained from the Semetex board shown in Figure 2.4. The PT timing is inherent in the original hardware and the PC timing is obtained through thesis software.

DA00-DA06 were chosen to function as the Y addresses and DD0-DD6 were chosen to function as the X addresses; DD7 to generate SPA7; WRITE for PT; and PRINT for PC. The board was set to the hexadecimal offset segment address B000 by properly setting the eight switches of the interface board. The high order bit is on the right and a switch in the down position represents a binary one. Any data sent to the Semetex board would have to be offset by this segment (B000).

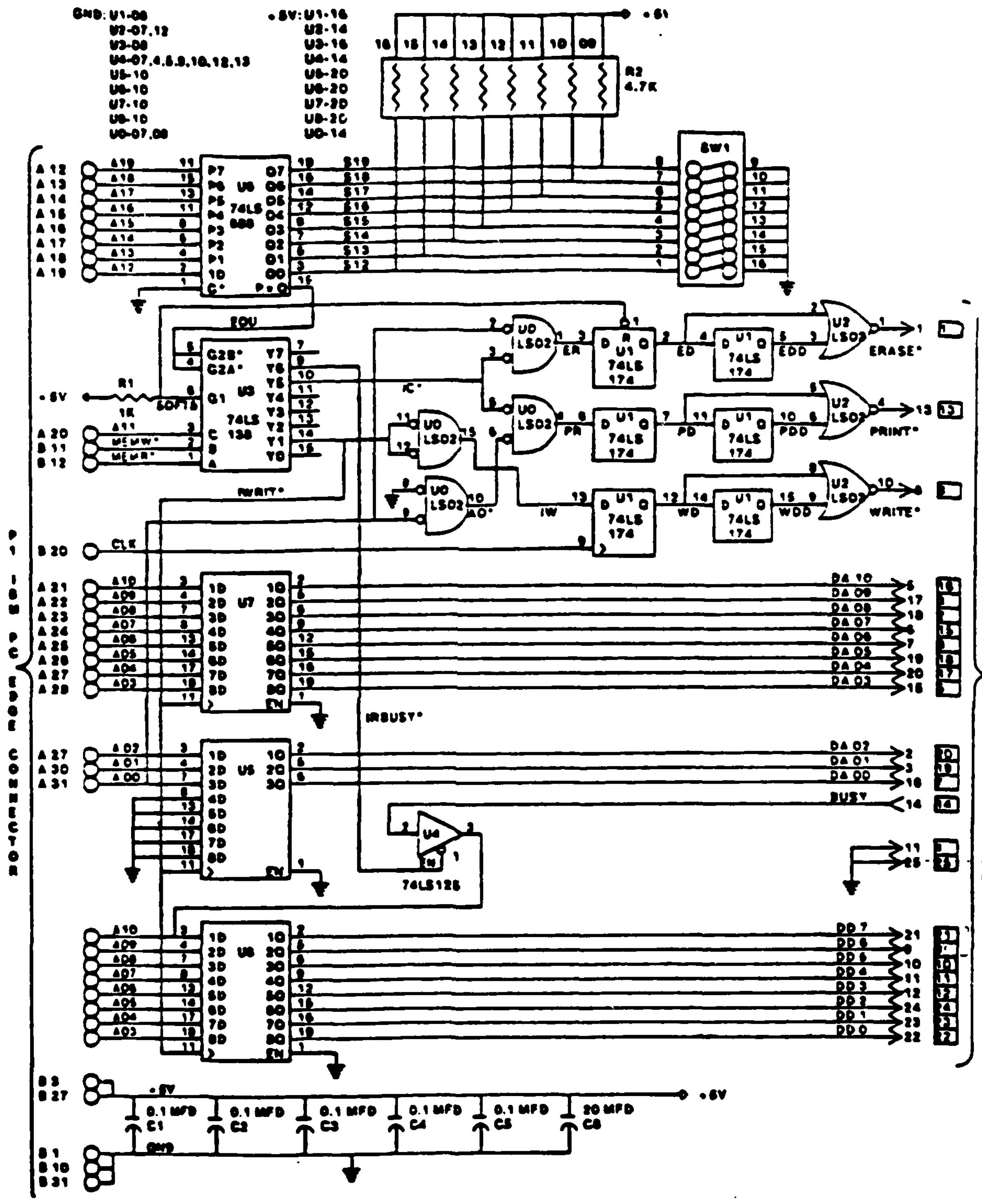


Figure 2.4. Semetex interface board schematic diagram (18).

Pixel selection is accomplished by sending the desired X-Y address to DA00-DA06 and DD0-DD6. For example, a write to decimal 50, 100 would nucleate the pixel at column 50, row 100 (a write is accomplished in the software by poking data to an address ⟶ software is explained in Appendix A). DD7 (SPA7 on the Litton X driver card) has to be high (binary one) to turn the pixel on. Before writing to the desired location, a logical OR of the row (DD0-DD6) and hex 80 (binary 1000 0000) must occur. The logical OR assures that DD7 is high without changing the data in DD0-DD6. If the pixel is to be turned off, a simple write to column,row is sufficient since DD7 is low (binary zero) by default.

Any write to column, row triggles a WRITE pulse since WRITE is tied to DA00-DA06 through flip-flops and the clock. A write to decimal offset 2049 after nucleation of all desired pixels will produce the PRINT pulse, sending current through the coil and completing the pixel write/erase process. A summary of pixel write/erase operation is shown in Figure 2.5. The C1 cable connection between the Semetex interface board and the Litton X driver card is shown in Table 2.2.

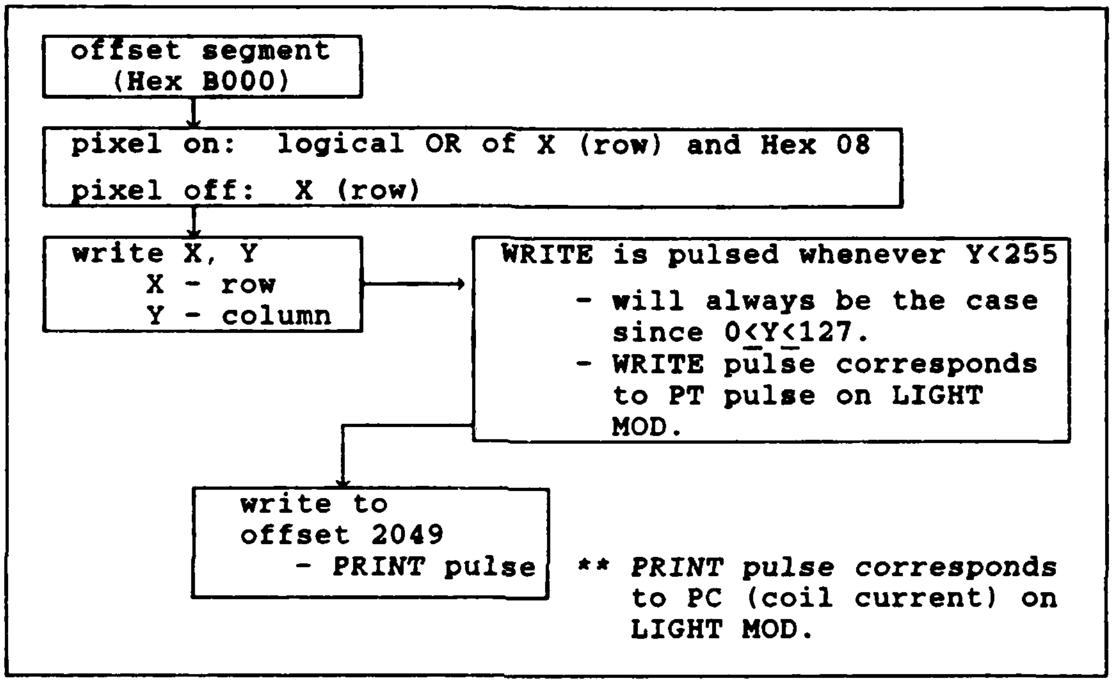

offset segment (Hex B000)
pixel on: logical OR of X (row) and Hex 08
pixel off: X (row)
write X, Y
X - row
Y - column
WRITE is pulsed whenever Y<255
- will always be the case since 0≤Y≤127.
- WRITE pulse corresponds to PT pulse on LIGHT MOD.
write to offset 2049
- PRINT pulse
** PRINT pulse corresponds to PC (coil current) on LIGHT MOD.

Figure 2.5. Pixel write/erase control through Semetex interface board to LIGHT MOD device.

Table 2.2
Cable Assembly C1
Semetex Interface Card → X Driver Card

| Semetex interface card | | Litton X driver card | | |
|---|---|---|---|---|
| pin # | Signal name | pin # | Signal name | Cable signal function |
| 7 | DA00 | 1 | FPB0 | X Address LSB |
| 19 | DA01 | 2 | FPB1 | " |
| 20 | DA02 | 3 | FPB2 | " |
| 5 | DA03 | 4 | FPB3 | " |
| 17 | DA04 | 5 | FPB4 | " |
| 18 | DA05 | 6 | FPB5 | " |
| 6 | DA06 | 7 | FPB6 | " MSB |
| 22 | DD0 | 9 | FPA0 | Y Address LSB |
| 23 | DD1 | 10 | FPA1 | " |
| 24 | DD2 | 11 | FPA2 | " |
| 12 | DD3 | 12 | FPA3 | " |
| 11 | DD4 | 13 | FPA4 | " |
| 10 | DD5 | 14 | FPA5 | " |
| 9 | DD6 | 15 | FPA6 | " MSB |
| 21 | DD7 | 17 | SPA7 | Write/Erase Control |
| 4,25 | GND | 20 | GND | Ground |
| 14 | BUSY | 19 | GND | Ground |
| 8 | WRITE | 21 | PT | XY Pulse |
| 13 | PRINT | 22 | PC | Coil Pulse |

# III. Polar-Logarithmic Coordinate Transformation CGH for Scale and Rotation Invariance

In this chapter, the scale and rotation invariant polar-logarithmic (lnr-$\theta$) coordinate transformation (CT) is used to achieve an in-plane distortion invariant feature space. The CT is performed by a computer generated hologram developed on a laser printer and a transforming lens.

The first section develops the necessary equations to produce the CT CGH. The results obtained by placing the CGH in the input plane of an optical processor are presented and discussed in the final part of this chapter. Appendix B presents the fabrication method used to develop the CGH.

## Design of the Coordinate Transformation CGH

The coordinate transformed image appears in plane $P_2$ of Figure 3.1 when the CGH is placed directly against the input image in plane $P_1$ (6). The CGH has an ideal binary

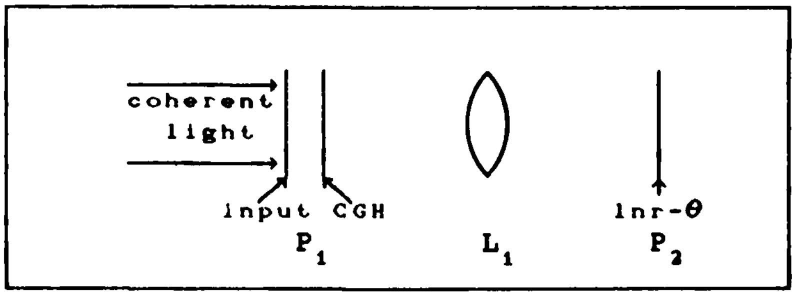


Figure 3.1. lnr-$\theta$ CT realization

grating-like pattern which simulates a continuous phase transmittance given by $h(x,y)=\exp[j\phi(x,y)]$, where $\phi(x,y)$ is the phase distribution. (The effect of the phase transformation exists in the first diffracted order of the grating).

The Fourier transform of the product of the input, $f(x,y)$, and transmittance, $h(x,y)$, is performed by lens $L_1$. In plane $P_2$ this product is defined mathematically by

$$F(u,v) = \iint_{-\infty}^{\infty} f(x,y)\exp[j\phi(x,y)] \\ *\exp[-j(2\pi/\lambda f_L)(xu+yv)dxdy \qquad (1)$$

where $\lambda$ is the wavelength of light used and $f_L$ is the focal length of lens $L_1$.

By taking the partial derivative of the Fourier kernel, it is found that the coordinate transformation which maps an input point $(x,y)$ to an output point $(u,v)$ is (7:3100):

$$x \longrightarrow u = \frac{\lambda f_L}{2\pi} \frac{\partial\phi(x,y)}{\partial x} \qquad (2)$$

$$y \longrightarrow v = \frac{\lambda f_L}{2\pi} \frac{\partial\phi(x,y)}{\partial(y)} \qquad (3)$$

To perform the $\ln r$-$\theta$ CT, the appropriate expression for $\phi(x,y)$ must be determined which satisfies (2) and (3). This transformation can be expressed by

$$u = \ln(x^2 + y^2)^{1/2} \qquad (4)$$

$$v = -\tan^{-1}(y/x) \qquad (5)$$

The proper phase function is found by using (4) in (2) and (5) in (3) then solving the following equations for $\phi(x,y)$ (7: 3099-3104).

$$\ln(x^2+y^2)^{1/2} = \frac{\lambda f_L}{2\pi} \frac{\partial\phi(x,y)}{\partial(x)} \tag{6}$$

$$-\tan^{-1}(y/x) = \frac{\lambda f_L}{2\pi} \frac{\partial\phi(x,y)}{\partial(y)} \tag{7}$$

The result is the continuous phase function

$$\phi(x,y) = [2\pi/\lambda f_L]*[\ln(x^2 + y^2)^{1/2} - y\tan^{-1}(y/x)-x] \tag{8}$$

The use of Equation (8) to form a binary grating pattern (CGH) to perform the lnr-$\theta$ coordinate transformation, the printing of that pattern on a laser printer, and the method used to reduce the CGH onto a high resolution film plate is discussed in Appendix B.

## Experiment and results

The final CGH used had dimensions of 10x10mm and a carrier frequency of 35 lines/mm . The CGH was placed in the input plane $P_1$ directly against the input image as shown in Figure 3.2. Input images consisted of various sized squares, circles, and letters. The laser source used was a 25mW HeNe ($\lambda$=6328nm) and the transform lens ($L_1$) had a focal length of 300mm. The experimental setup is shown in Figure 3.2.

The lnr-$\theta$ CT pattern was observed in plane $P_2$ using a Sony X-38 CCD (charge coupled device) camera and AT&T frame grabber on a Zenith 248 computer system. The lnr-$\theta$ CT is

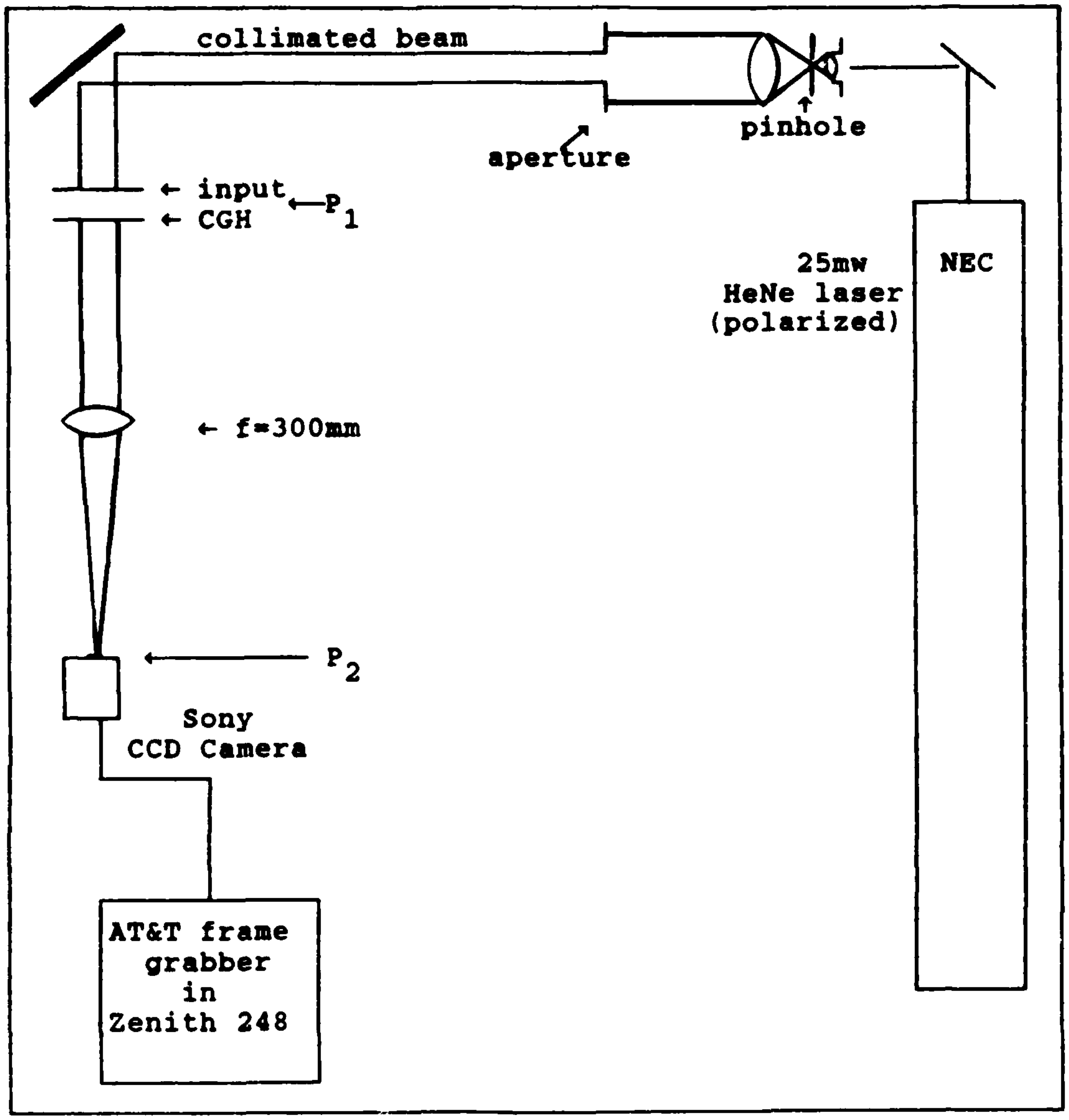


Figure 3.2. Scale and rotation invariant setup.

present in the first diffracted order in plane $P_2$. The angular seperation of diffracted orders is dependent on the carrier frequency of the grating pattern ($a$) and the wavelength of incident light ($\lambda$) by $\theta = \sin^{-1}(a\lambda)$. For $a$=35 lines/mm and $\lambda$=0.6328$\mu$m, the angular seperation, $\theta$, is 1.27

degrees. A diffraction pattern which is related to the pixel size of the CGH is also present. The three dots above the lnr-$\theta$ CT shown in the experimental results corresponds to this diffraction pattern.

The transformed patterns were observed to remain unchanged for each class of images (except for horizontal and vertical shifts). A rotation of the input image caused a horizontal (cyclical, along the $\theta$ axis) shift in the CT plane $P_2$. This cyclical shift is proportional to the input rotation (13). The $P_2$ transformed pattern was found to shift vertically (along the lnr axis) for input scale changes. These results verified the use of the CGH for the desired lnr-$\theta$ CT. The results are shown in Figures 3.3-3.8.

The experimental transformations fade out at small r relative to the theoretical due to the way light is transferred from the x-y plane to the lnr-$\theta$ plane. A small amount of light from a central polar region of the x-y plane is transferred to the same amount of area in the lnr-$\theta$ plane as a large amount of light from an outer polar region of the x-y plane.

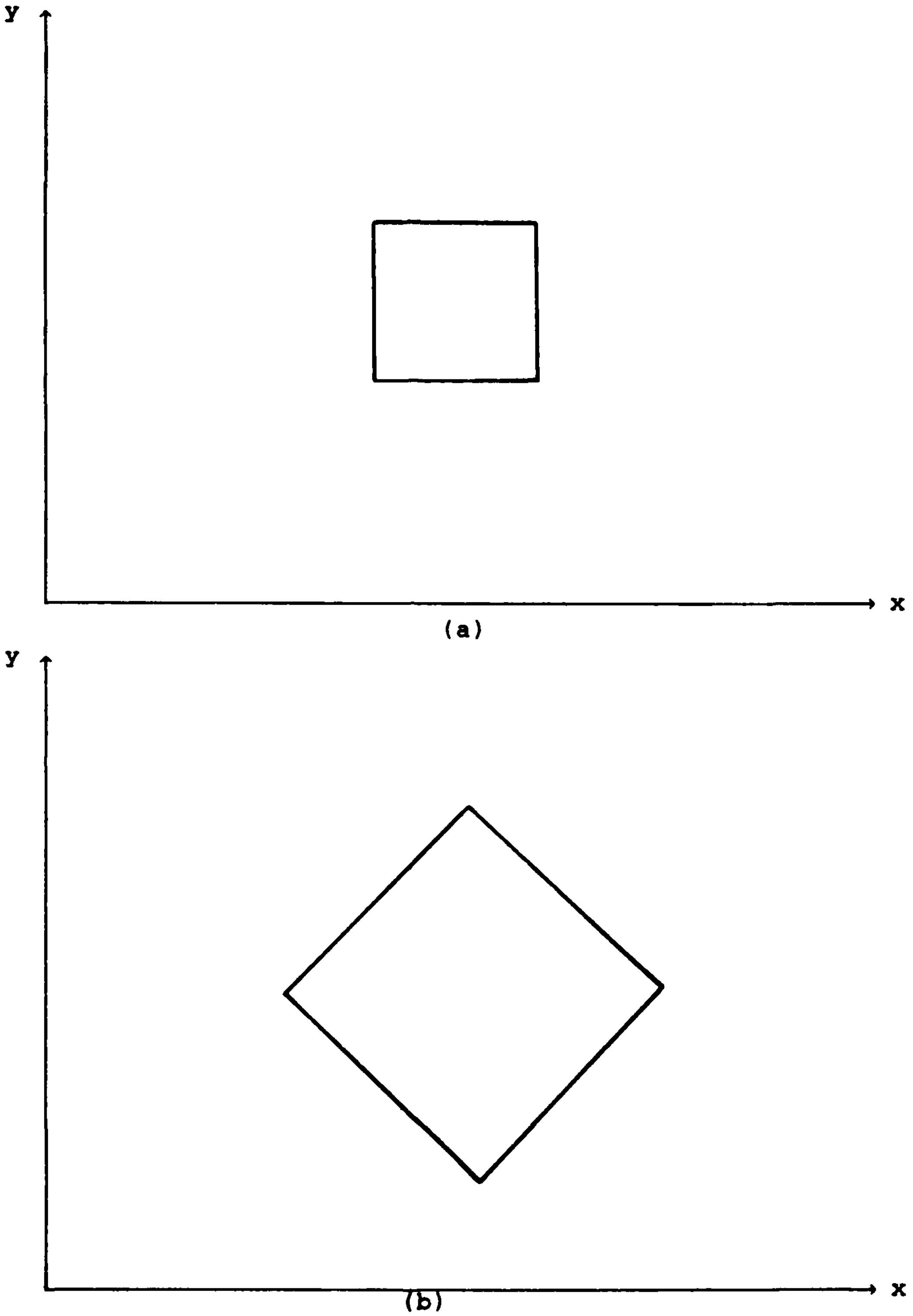
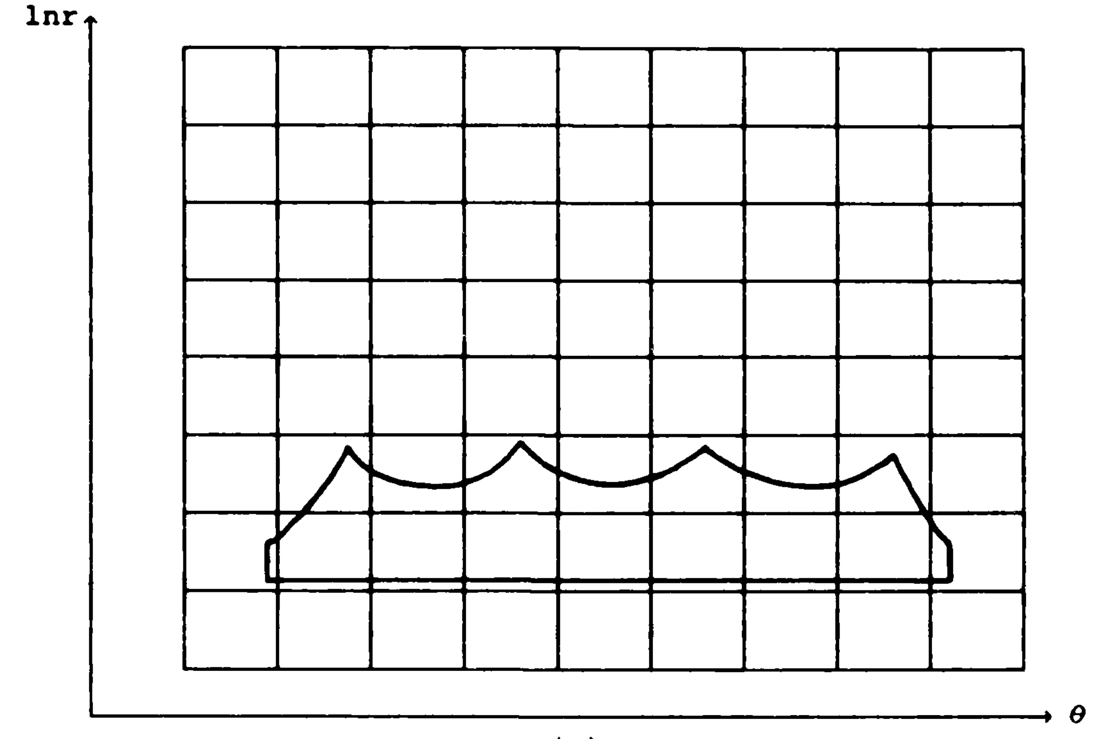
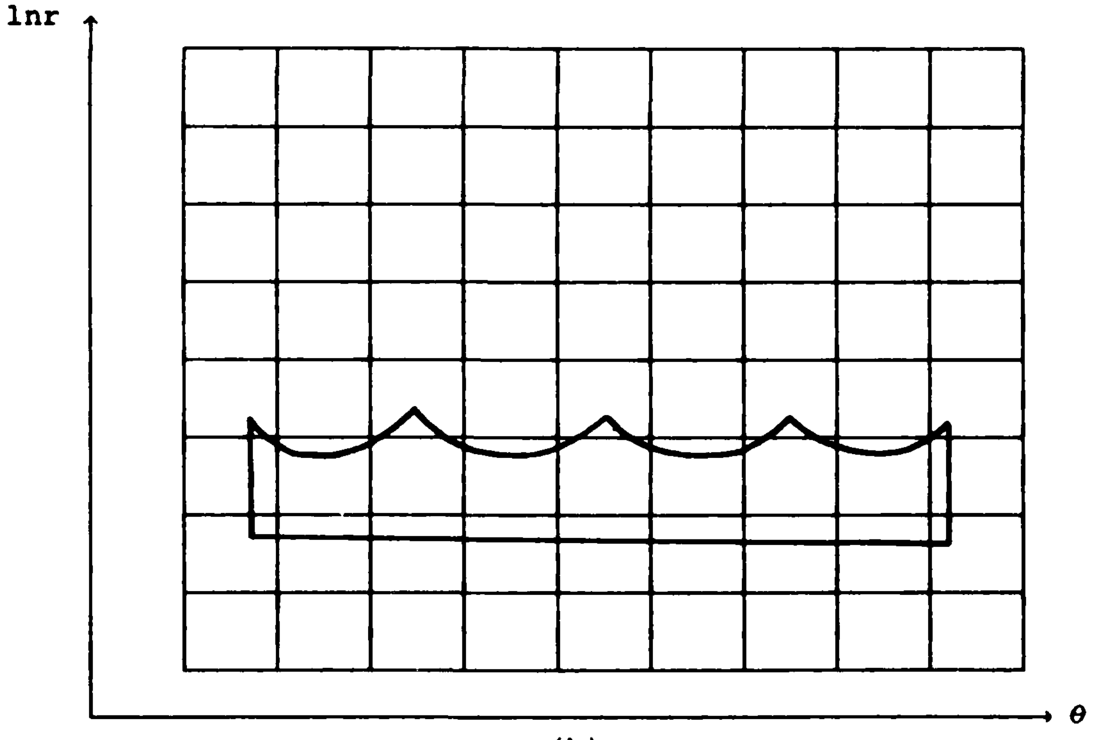


Figure 3.3. (a) Input square.
(b) 167% scale change and 45 rotation.

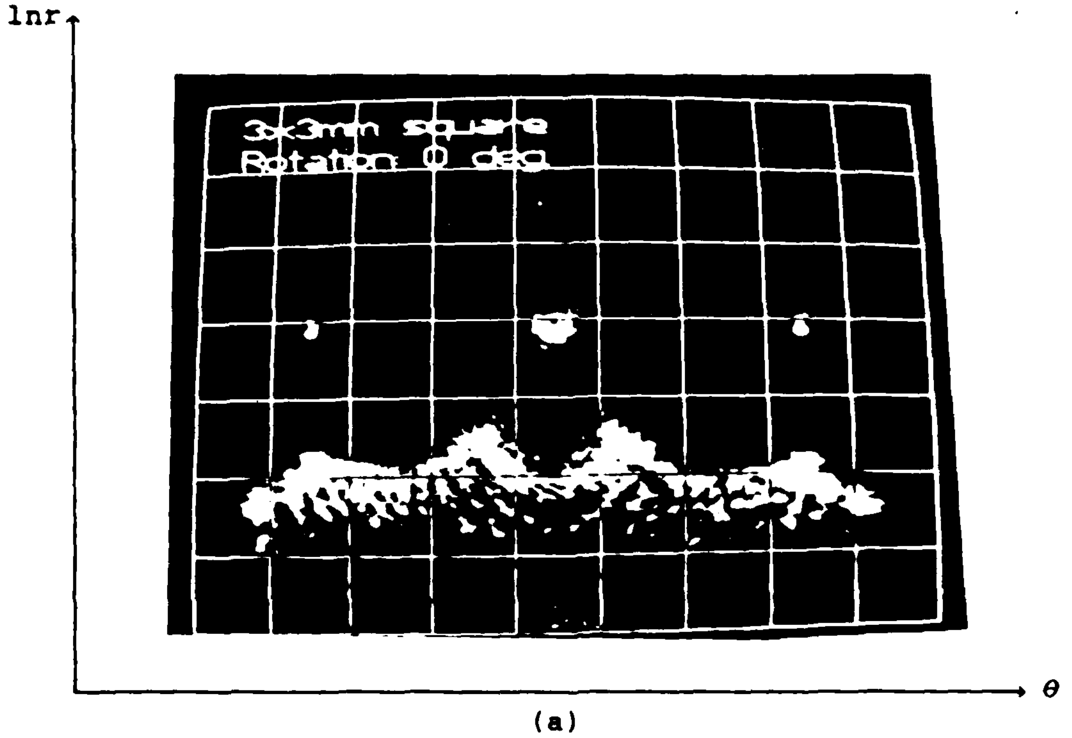


(a)



(b)

Figure 3.4. Theoretical lnr-$\theta$ CT of squares.



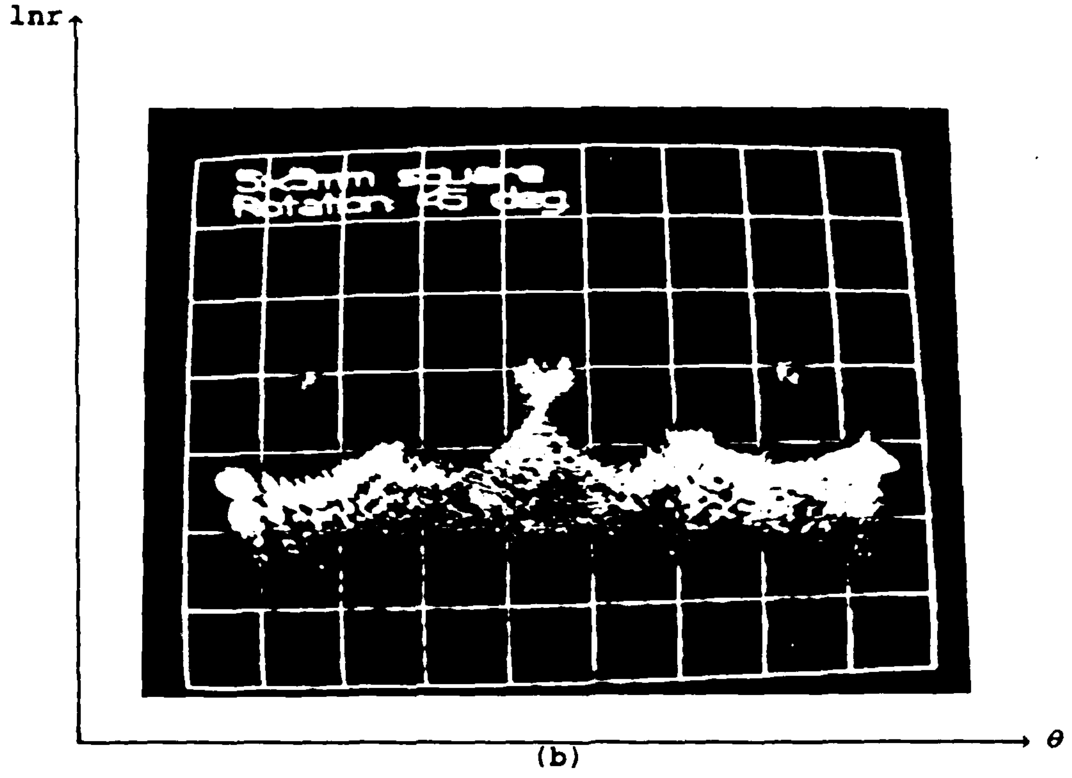
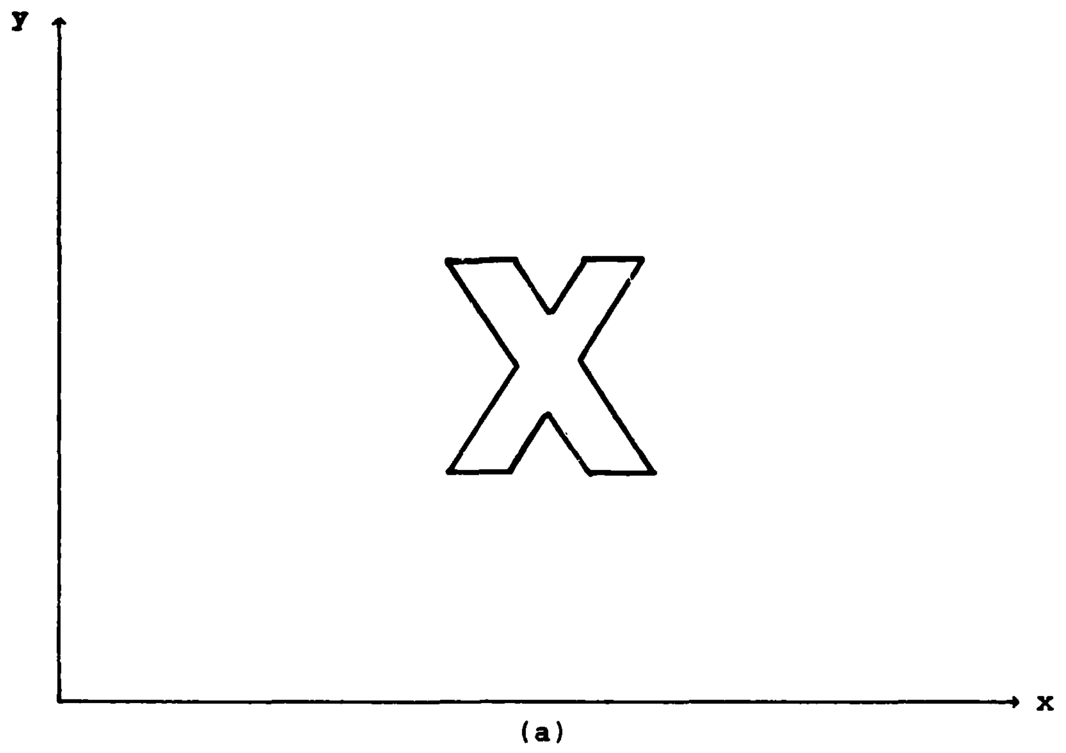


Figure 3.5. Experimental lnr-$\theta$ CT of squares.



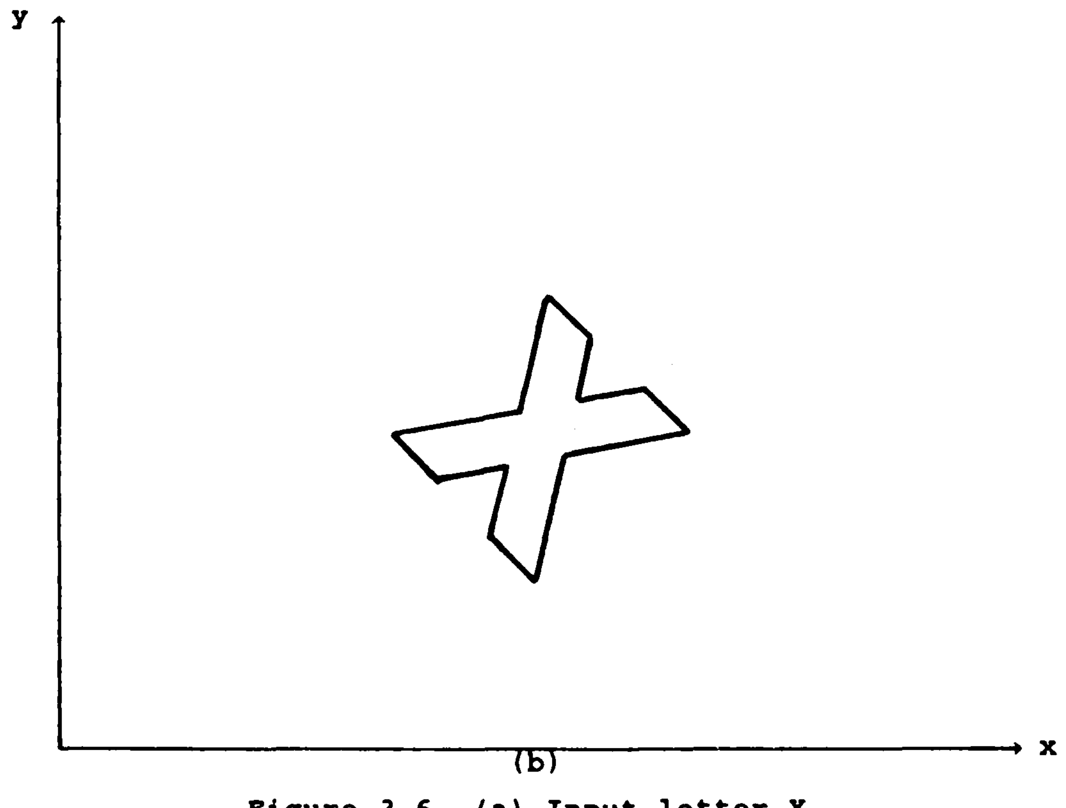
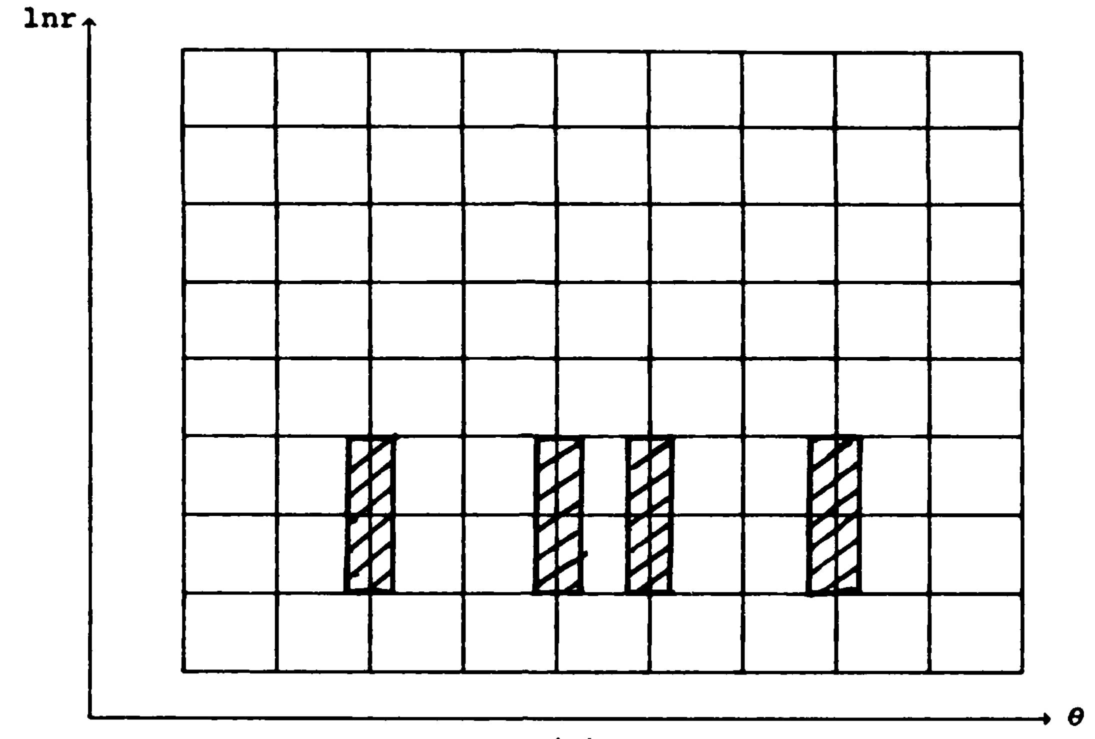
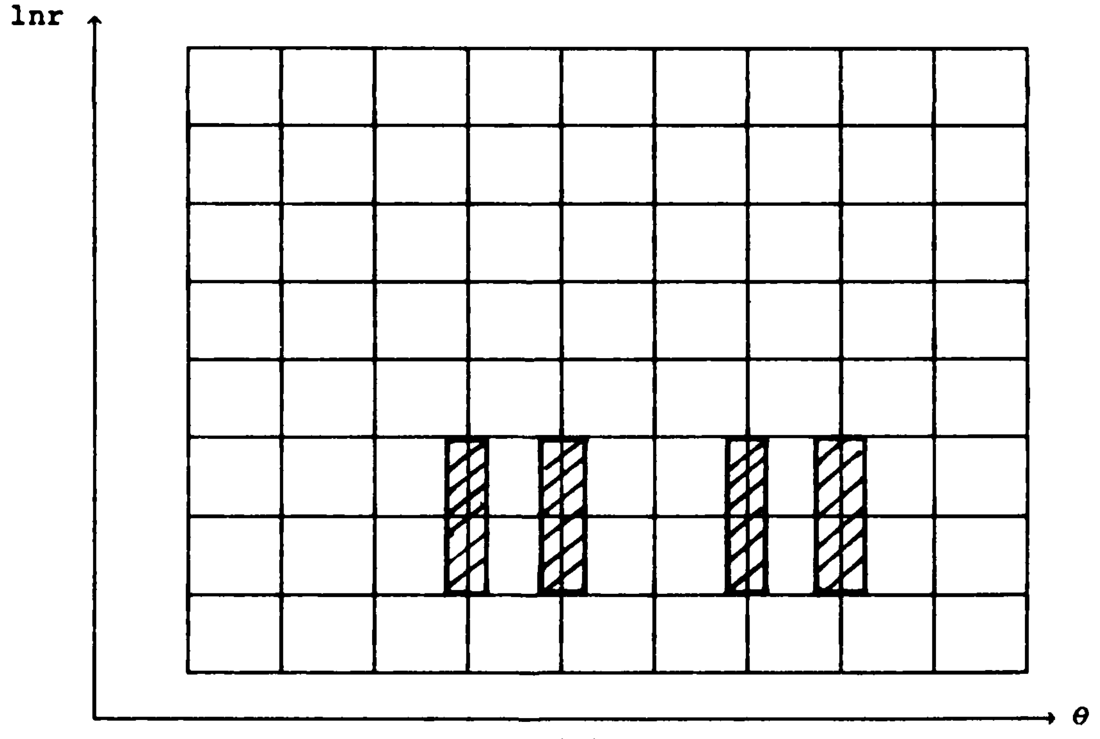


Figure 3.6. (a) Input letter X.
(b) 45 rotation of X.

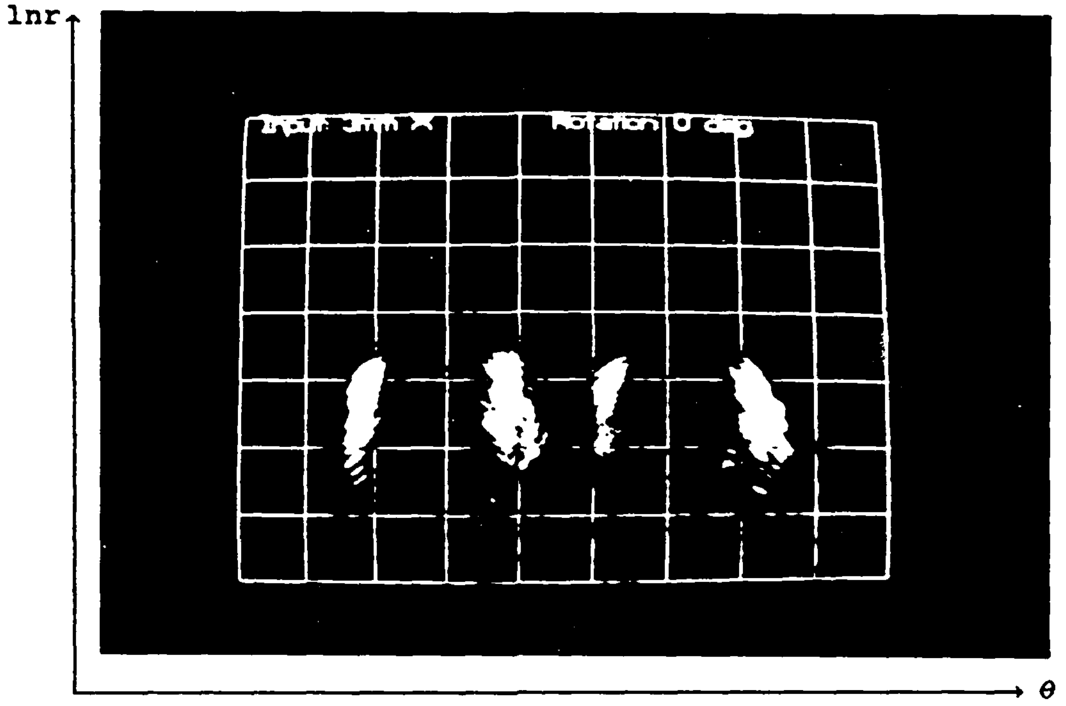


(a)

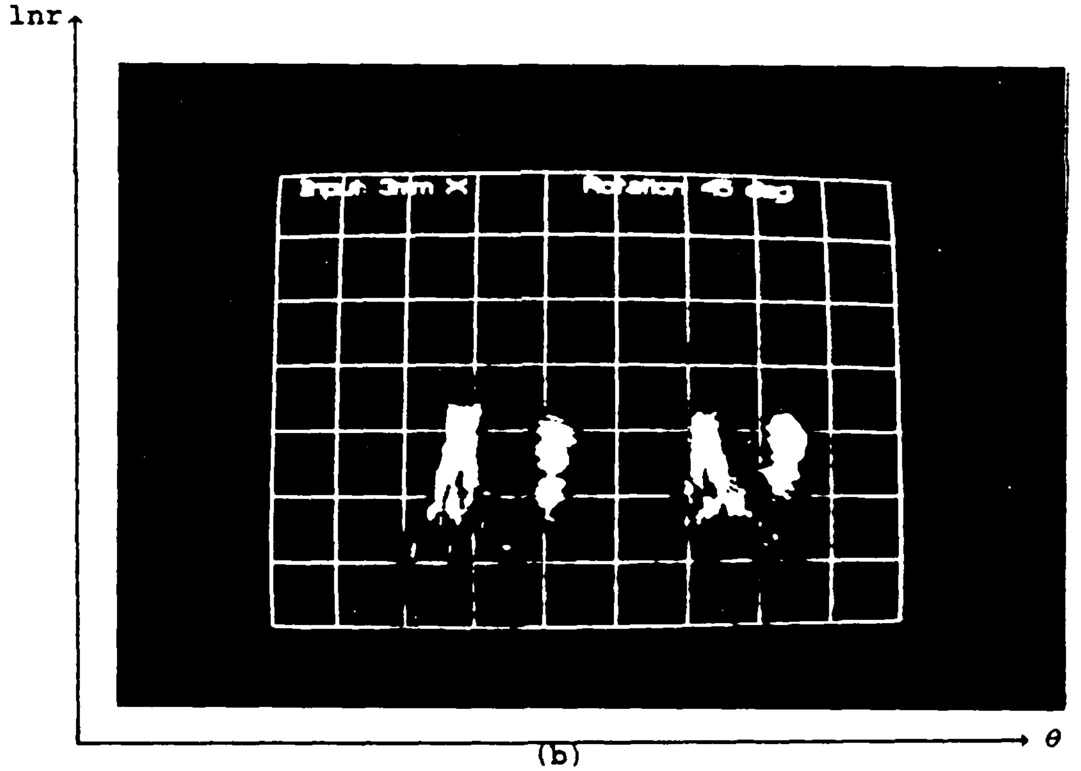


(b)

Figure 3.7. Theoretical lnr-$\theta$ CT of X's.



(a)



(b)

Figure 3.8. Experimental lnr-$\theta$ CT of X's.

## Conclusion

This chapter developed the necessary equations to form the lnr-$\theta$ coordinate transformation binary grating CGH. The results from using the CGH in the input plane of an optical processor to obtain scale and rotation (but not position) invariance was also presented. In order to become position invariant, the magnitude of the Fourier transform must be used as the input to the CGH. However, the CGH cannot be placed in the Fourier transform plane since the light is not a plane wave at that point. With non-parallel light entering the CGH, the lnr-$\theta$ CT plane becomes a diffracted mess with input images overlayed in a continuous fashion. To overcome this, the magnitude squared of the Fourier transform will be recorded using a CCD camera and displayed on the SLM. The SLM, with parallel light passing through it, will become the input of the CGH. Another option is to take a picture of the Fourier transform and place the developed slide into the system to act as the input to the CGH. The SLM will only give a thresholded binary image of the Fourier transform whereas the picture will contain the grey levels.

Chapter two described the modification and interfacing which was necessary to incorporate the LIGHT MOD into the optical processor as a SLM. Chapter 4 will then present the position, scale, and rotation invariant optical system using slides since the LIGHT MOD broke while trying to perform the experiments.

# IV. Position, Scale, and Rotation Invariant Feature Space

## Introduction

An ideal target recognition system should be able to detect the presence of every occurrence of a particular reference pattern in an input scene. However, six major distortions in the input pattern prevent a recognition system from working properly (14):

1. changes in target position,
2. changes in target size,
3. in-plane rotation of the target,
4. out-of-plane rotation of the target,
5. changes in target shape,
6. and changes in scene clutter.

This chapter investigates the first three distortions. Problem areas four, five, and six are much more difficult to compensate and were not investigated in this study.

The use of an optical correlator and matched spatial filter for target recognition is advantageous due to its ability to process in parallel. However, the conventional correlator is unable to recognize scaled and rotated versions of the reference object.

If two functions to be correlated are identical except for a scaling factor, the resulting correlation peak will not be as sharp as when the scaling factor is one. Likewise, a rotation of one of the functions will quickly degrade the correlation peak (2:1652; 15).

Since an optical correlator was to be used to determine whether a target in the input scene matched a reference target, the input scene and template had to be transformed into a position, scale, and rotation invariant (PSRI) feature space.

A PSRI feature space will be developed in the first section of this chapter. Three transformations were performed on the template and input scene in order to obtain a PSRI feature space. A Fourier transform and a lnr-$\theta$ coordinate transformation (CT) was used to change any scale and rotation difference into shifts along the feature space axes. The next section of the chapter shows the optical system which transforms an input image into a PSRI feature space. The PSRI feature space can then be used as the input to an optical correlator, as will be shown in Chapter 5. The final section presents results of the optical transformation which develops the PSRI feature space.

It should be noted that the optical PSRI feature space implemented in this thesis was based on the digital electronic algorithm developed by Kobel and Martin on the VAX computer system at AFIT (14). Many of the scenes and templates used in the computer algorithm were placed on film and used as the inputs of the optical system. Comparisons to the digital electronic algorithm will be made where applicable.

## Position Invariance

Position invariance is accomplished by using the magnitude of the Fourier spectrum, $|I(f_x,f_y)|$. The Fourier transform (FT) shift theorem will show position invariance (10:9):

$$\text{If} \quad F\left\{i(x,y)\right\} = I(f_x,f_y), \quad \text{then}$$

$$F\left\{i(x-\alpha,y-\beta)\right\} = I(f_x,f_y)\exp\left[-j2\pi(\alpha f_x+\beta f_y)\right] \tag{9}$$

where

| | |
|---|---|
| $i(x-\alpha,y-\beta)$ = | input image shifted by $\alpha$-units in the x-direction and $\beta$-units in the y-direction. |
| $F\{\quad\}$ = | two-dimensional Fourier Transform. |

"This simply states that a shift in the input scene contributes a linear phase factor to the Fourier transform but does not impact the magnitude of the Fourier transform " (14:17). Therefore, by using the magnitude (or magnitude squared of the Fourier spectrum) there are no variations due to a shift (relocation of target) in the input scene plane. the magnitude squared of the FT was used in the optical system since the intensity of the FT was detected.

## Scale and Rotation Invariance

The rotation of an object in the input scene plane will result in an identical rotation of the magnitude of the object's FT. The rotation of the magnitude spectrum in a rectangular coordinate system can be transformed into a linear shift by mapping the magnitude spectrum to a polar coordinate system as shown in Chapter 3. The magnitude spectrum, $|I(f_x,f_y)|$, will now be in the form $|I(f_r,f_\theta)|$, with spatial frequencies in the angular ($f_\theta$) and radial ($f_r$) directions. The new spatial frequency coordinates are given by (14:17):

$$f_\theta = \tan^{-1}\left[\frac{f_y}{f_x}\right] \tag{10}$$

and

$$f_r = \sqrt{f_x^2 + f_y^2} \tag{11}$$

Any rotation will now appear as a linear shift of the polar magnitude spectrum along the angular axis, $f_\theta$.

A logarithmic scaling of the radial axis of the polar magnitude spectrum plane is then required for scale invariance. "The Fourier transform 'similarity' theorem states that a magnification or shrinkage of the coordinates in the space domain corresponds to a respective shrinkage or magnification of the coordinates in the frequency domain" (14:18). The similarity theorem is expressed by (10:9):

$$F\left\{ i(x/\alpha,\ y/\beta) \right\} = |\alpha\beta| I(\alpha f_x, \beta f_y) \tag{12}$$

where $\alpha$ and $\beta$ are the scaling constants for the x and y coordinates.

The spatial frequencies $f_x$ and $f_y$ are uniformly scaled when $\alpha=\beta$ (that is, the object is uniformly scaled in the x and y-direction). As a result, $\left[ f_x / f_y \right] = \left[ \alpha f_x / \beta f_y \right]$. Therefore, the angular axis ($f_\theta$) of the polar magnitude spectrum plane will not be affected by a uniform scaling due to the ratio $\left[ f_x / f_y \right]$ in Equation (10).

By scaling the radial frequency axis in Equation (11) by $\alpha$

$$f_r = \sqrt{\alpha^2 f_x^2 + \beta^2 f_y{}^2} \tag{12}$$

and taking the natural logarithm yields:

$$\begin{aligned} f_r &= \ln\left[ \sqrt{\alpha^2 f_x^2 + \beta^2 f_y{}^2} \right] \\ &= (1/2)\ln\left[ \sqrt{\alpha^2 f_x^2 + \beta^2 f_y{}^2} \right] \\ &= \ln(\alpha) + (1/2)\ln\left( f_x^2 + f_y^2 \right) \end{aligned} \tag{13}$$

The second term of Equation (13) is just the natural logarithm of the unscaled radial spectrum. This shows that a uniform scaling of an image by $\alpha$ will cause a linear shift of the amount $\ln(\alpha)$ along the logarithmic radial frequency axis.

An optical scale and rotation invariant feature space was developed and shown in Chapter 3 using the lnr-$\theta$ CT CGH and a transforming lens. That same processor was incorporated into the PSRI system by placing the magnitude squared of the FT against the lnr-$\theta$ CT CGH.

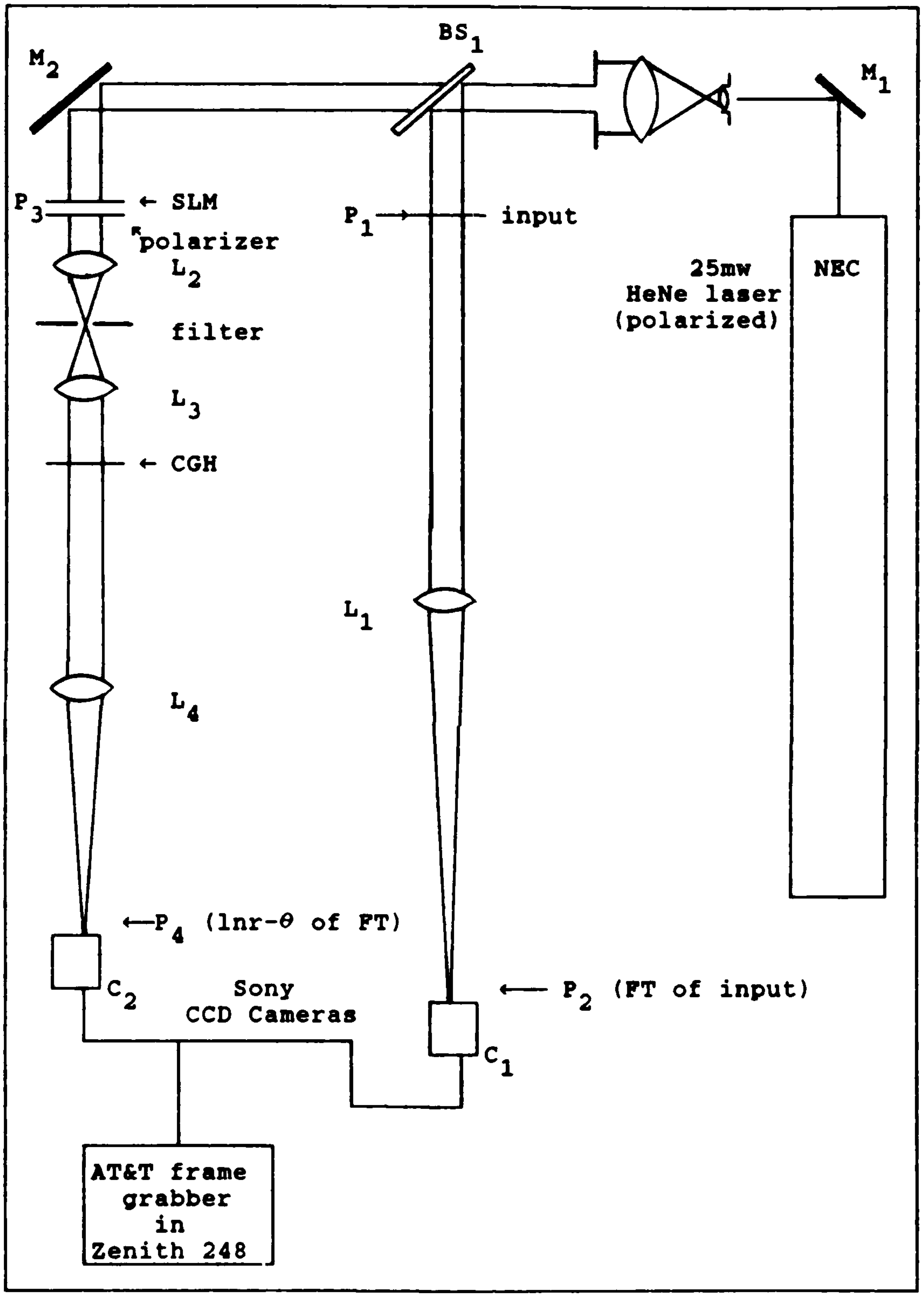


Figure 4.1. PSRI feature space optical processor.

## Optical Processor for PSRI Feature Space

The experimental set-up used to transform an input scene or template from an x-y system to a PSRI feature space is shown in Figure 4.1.

The first step in the optical processor accomplished position invariance. The template or scene was placed in the input plane $P_1$ and Fourier transformed by lens $L_1$. The magnitude squared of the FT ($|FT|^2$) was detected in plane $P_2$ by a Sony CCD camera. The $|FT|^2$ remained unchanged when the target was relocated (shifted, but not rotated or scaled) in the input plane; thus giving the position invariance.

It was necessary to place a high pass filter adjacent to plane $P_2$ in order to obtain any information from the $|FT|^2$. The dc and lower order harmonics would cause the CCD camera to wash out since they were more intense than the higher frequency components. Plus, there is no real individual target characteristic information in the lower harmonics so it was not necessary to include them. Kobel and Martin (14:31-33) used a bandpass mapping function to exclude undesirable low and high spatial frequencies. They established an $f_{min}$ and $f_{max}$ by determining which portion of the spatial frequency spectrum was generated by the input target silhoette. They evaluated the size relationships of the aircraft template silhoette to the overall template window and the size relationship between the possible target and scene window. The lower bound ($f_{min}$) was determined by the targets largest physical feature and the upper bound

($f_{max}$) was determined by the targets smallest physical feature. They found that the fundamental harmonic of the largest physical feature (for a "typical" target) corresponded to the 4th harmonic whereas the fundamental harmonic of the smallest physical feature was the 55th harmonic. Therefore, they used a bandpass with $f_{min}=4$ and $f_{max}=55$. The results they obtained during correlation was very good so I suspected the same would be true using the optical processor. A filter which blocked the first 4 harmonics of the $|FT|^2$ would be the most desirable.

The program TARGASLM was then used to capture the $|FT|^2$ of the input target and transfer it to the SLM in plane $P_3$ where the $|FT|^2$ would be the input to the $\ln r$-$\theta$ CT processor. A picture of the $|FT|^2$ was also taken from the high resolution monitor using a 35mm camera with 200 speed color slide film. Since the SLM was only able to display a binary sampled version of the $|FT|^2$, the slide would capture the complete gray scaled image (limited by the resolution of the CCD detector and film) but using the slides would negate the real-time capability of the system. It was during these experiments that the SLM failed so all results shown at the end of this chapter used the 35mm slide technique.

The scale and rotation invariant ($\ln r$-$\theta$) optical processor used the $|FT|^2$ as the input to the $\ln r$-$\theta$ CT CGH. Lens $L_2$ and a low pass filter were used to filter out the harmonics of the SLM which were due to the repeated electrode structure of the SLM device. A very fine grid pattern is formed by the 128x128 crossed electrodes which created a large sinc pattern in both the x and y directions. Only the central harmonic needed to be passed since most of the displayed image is contained in it. Lens $L_3$ imaged the $|FT|^2$ onto the center of the CGH. The CGH, along with lens $L_4$, then performed the desired $\ln r-\theta$ CT on the $|FT|^2$ of the input target and the resulting PSRI feature space was generated in plane $P_4$. The magnitude squared of the PSRI feature space was detected by camera $C_2$ where it could then be transferred to the input of an optical correlator. The optical correlator is discussed in Chapter 5. A discussion of the experiment and the results are presented in the next section.

## Experimental results

The goal of these experiments was to transform an input template or scene into a PSRI feature space. With such a featue space, a target shifted to another part of the scene would have the same shape after transformation as the template. Any rotation or scale change of the target with respect to the template would relate to a simple horizontal or vertical shift in the PSRI feature space. Templates consisted of geometrical shapes, letters, and aircraft silhoettes. The aircraft silhoettes were those used by Kobel and Martin in their digital electronic computer algorithm. They were obtained from video files stored on the VAX computer system and placed on 200 speed 35mm slides to be used as inputs to the optical system. The real scenes containing the airplanes were also obtained from the computer system.

The first set of experiments consisted of placing simple geometric objects in the input plane $P_1$ and performing the FT with lens $L_1$. By using the high pass filter in front of camera $C_1$, quality FT's were observed in plane $P_2$. TARGASLM was then used to capture the $|FT|^2$ in video memory and transfer the $|FT|^2$ to the SLM in plane $P_3$ to function as the input to the CGH. Best results were obtained when only the center 256x256 of the 512x400 video image was sent to the SLM. The threshold value remained as a variable input to the user since the amount of energy passing through the input varied greatly. At the time an 8mW laser was used as the source and the CCD camera could barely detect the PSRI feature space in the diffracted order of the CGH/lens $L_4$ output ($P_4$). The system appeared to work properly but the SLM failed irremedially before any useful results were obtained after incorporating the 25mW laser into the system as the source.

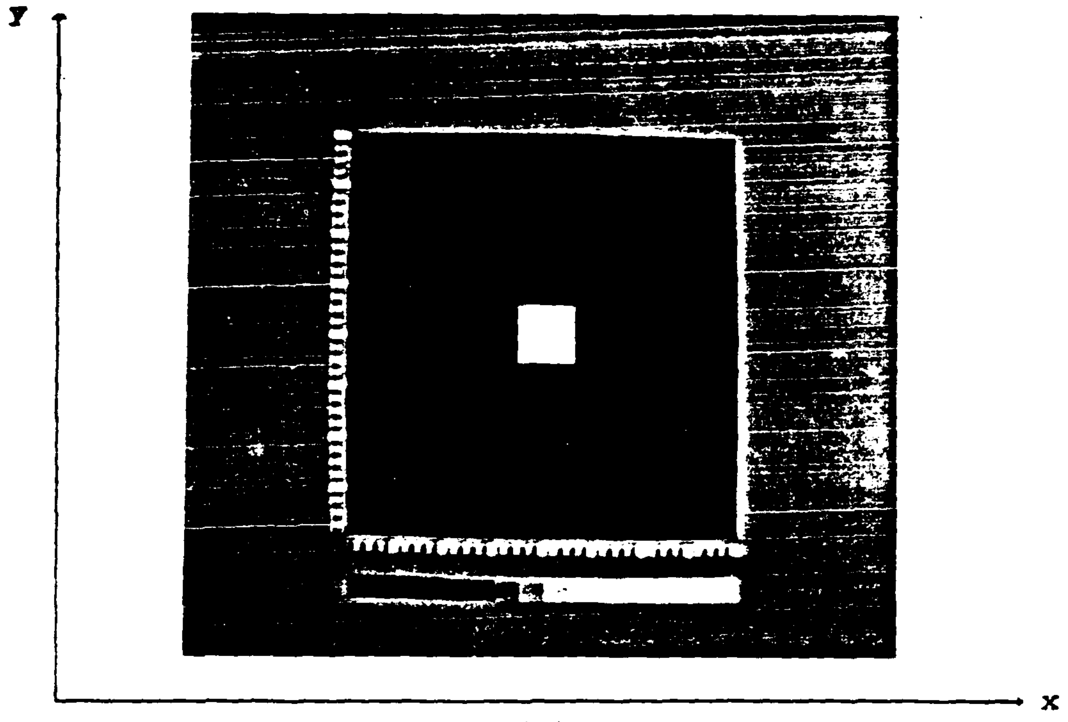
The first set of results consisted of a small square and a larger, rotated square as the inputs (Figure 4.2). The $|FT|^2$ of these two inputs are shown in Figure 4.3. The $|FT|^2$ were also placed on slides and inserted into plane $P_3$ where they were imaged onto the center of the lnr-$\theta$ CGH. The resulting PSRI feature space detected in plane $P_4$ is shown in Figure 4.4. Note the vertical (along the lnr axis) and horizontal ($\theta$ axis) shifts associated with the scale and rotation difference between the two inputs. With Figure 4.4a used as the reference template (matched spatial filter) and Figure 4.4b used as the input to an optical correlator, it is presumed that a very sharp correlation peak would result after a crosscorrelation is performed.

In the next set of results presented in this Chapter, the side view silhoette of an F15 was used as the template and a scaled and rotated version of the F15 was used as a comparison (Figure 4.5). Also included with the analysis was an F15 amongst some clouds in a real scene (Figure 4.6).

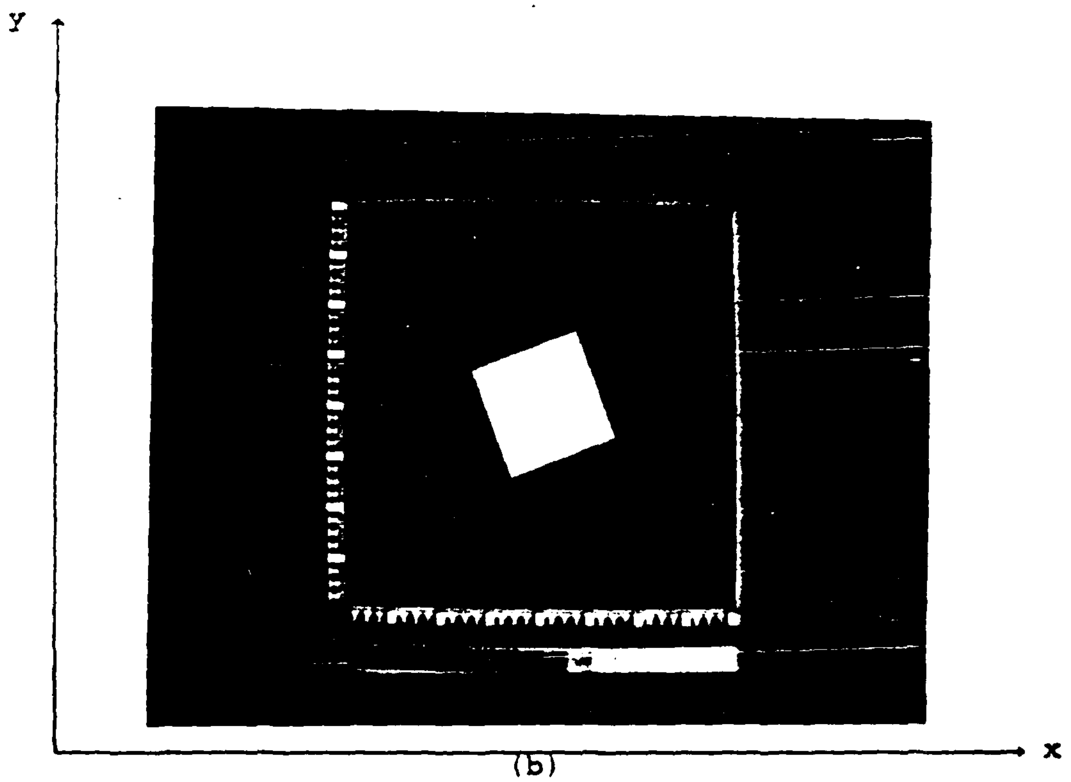
The PSRI feature spaces for both the templates and the real scene are shown in Figure 4.7. Again, this feature space shown could be used as an input to an optical correlator since each only differs by a horizontal or vertical shift.

When the airplane templates and real scenes were used, the results were not as clear as those obtained using simple geometrical objects. This can be attributed to three observed problem areas. The first, and most obvious, is that the objects are more complicated making the results less intuitive (there is really no a priori knowledge of how the PSRI feature space should look).

The next problem concerned the detection of the $|FT|^2$ after passing through the high pass filter. The filter consisted of an ink blot in the middle of a glass slide.

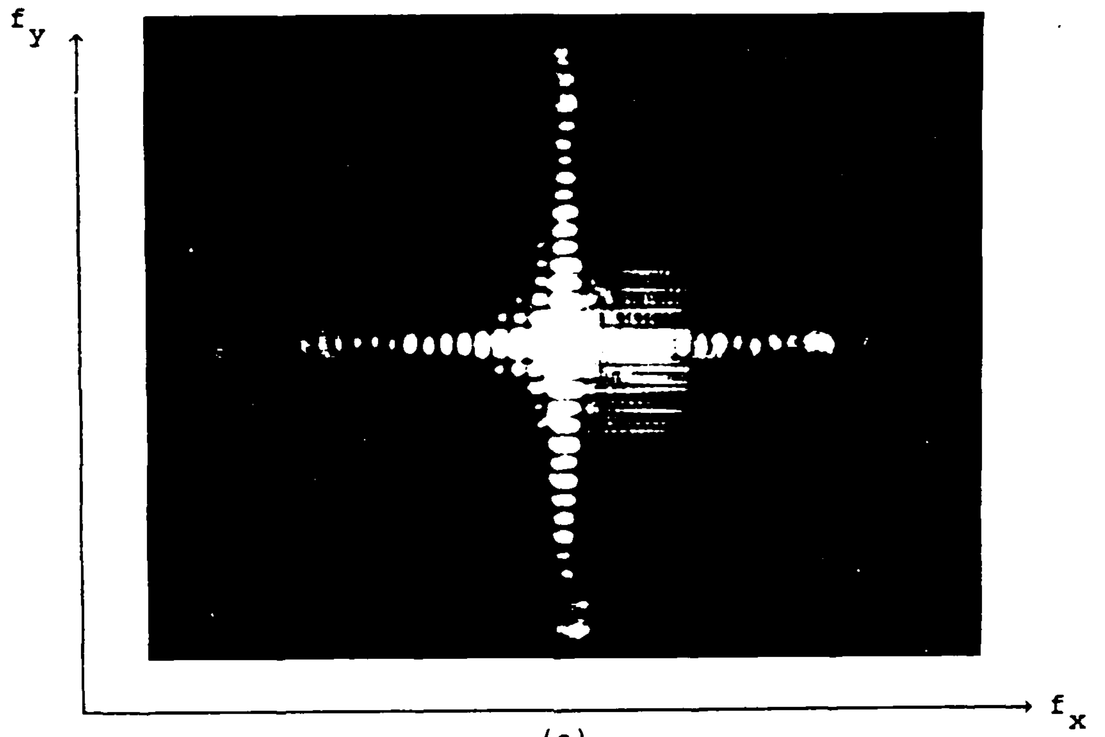


(a)

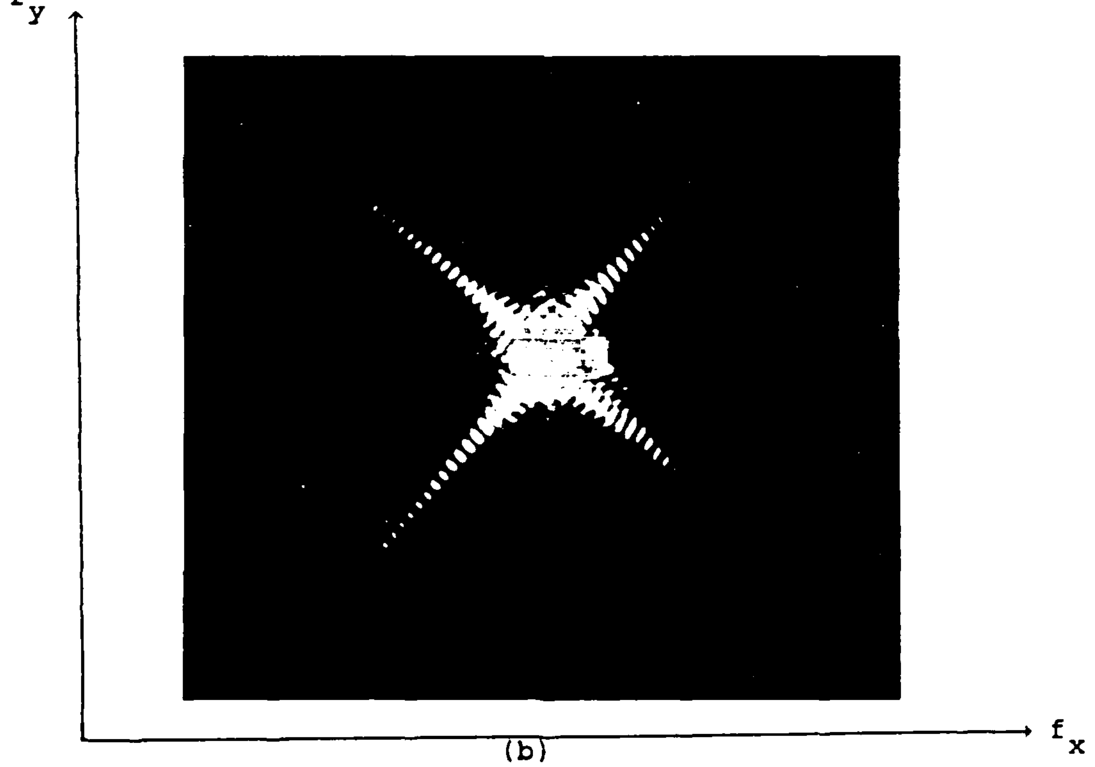


(b)

Figure 4.2. (a) Input square.
(b) Scaled and rotated square.

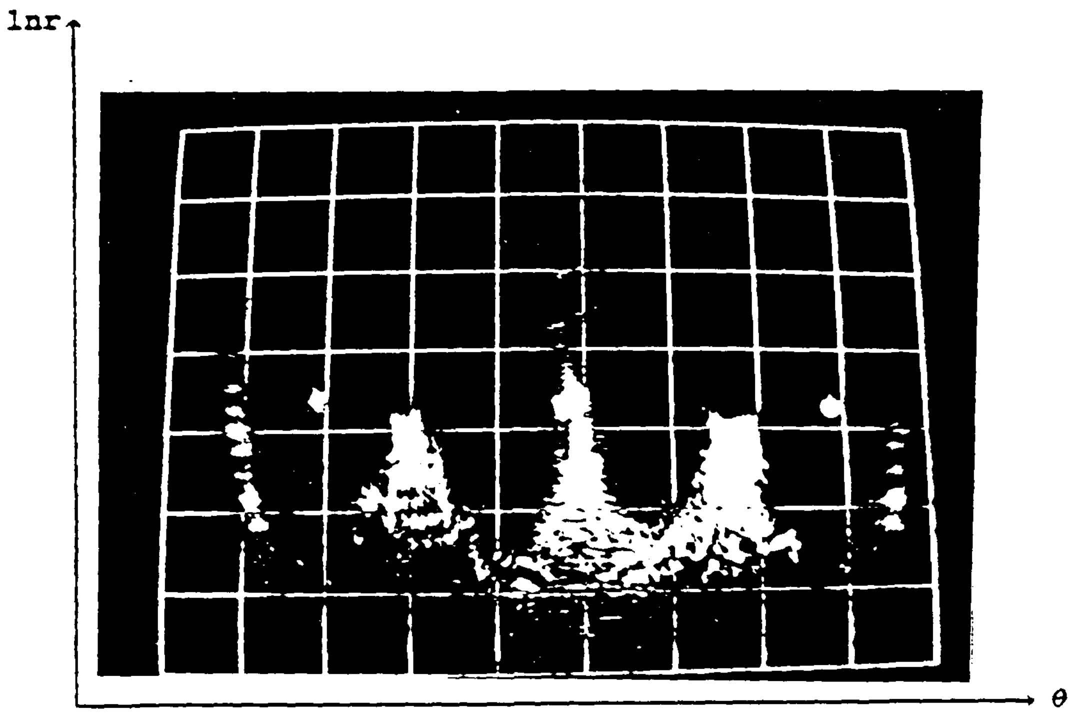


(a)

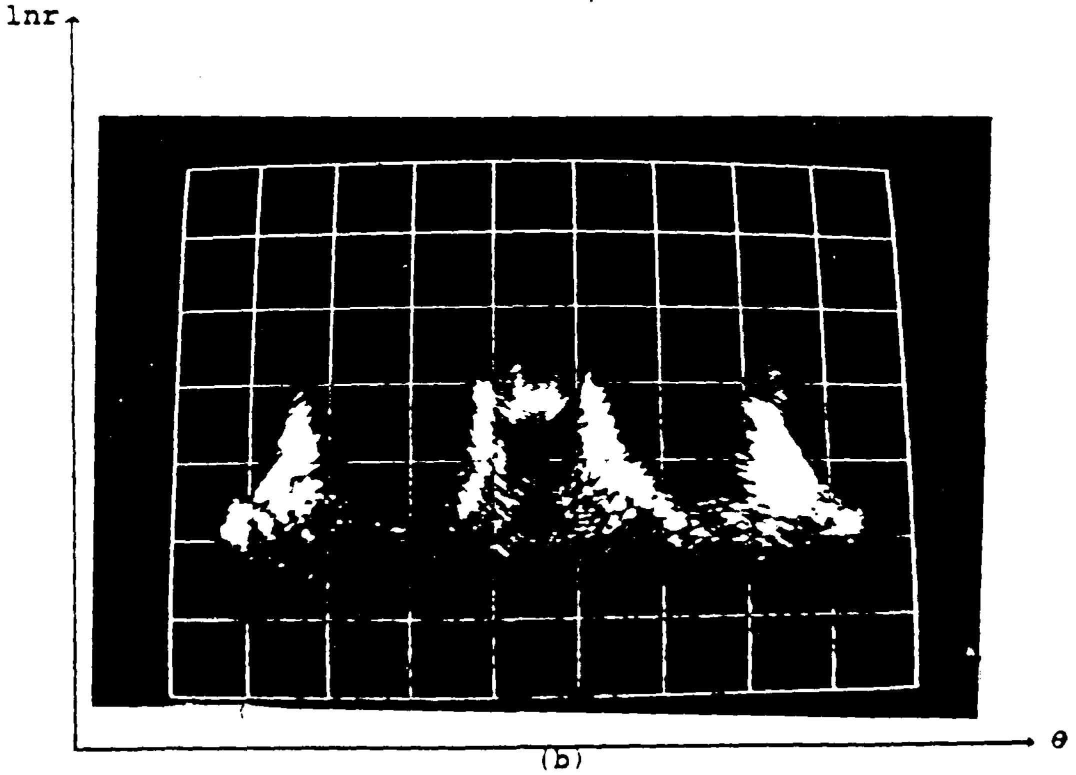


(b)

Figure 4.3. (a) FT of square.
(b) FT of scaled and rotated square.



(a)



(b)

Figure 4.4. PSRI feature space. (a) Input square (b) Scaled and rotated square.

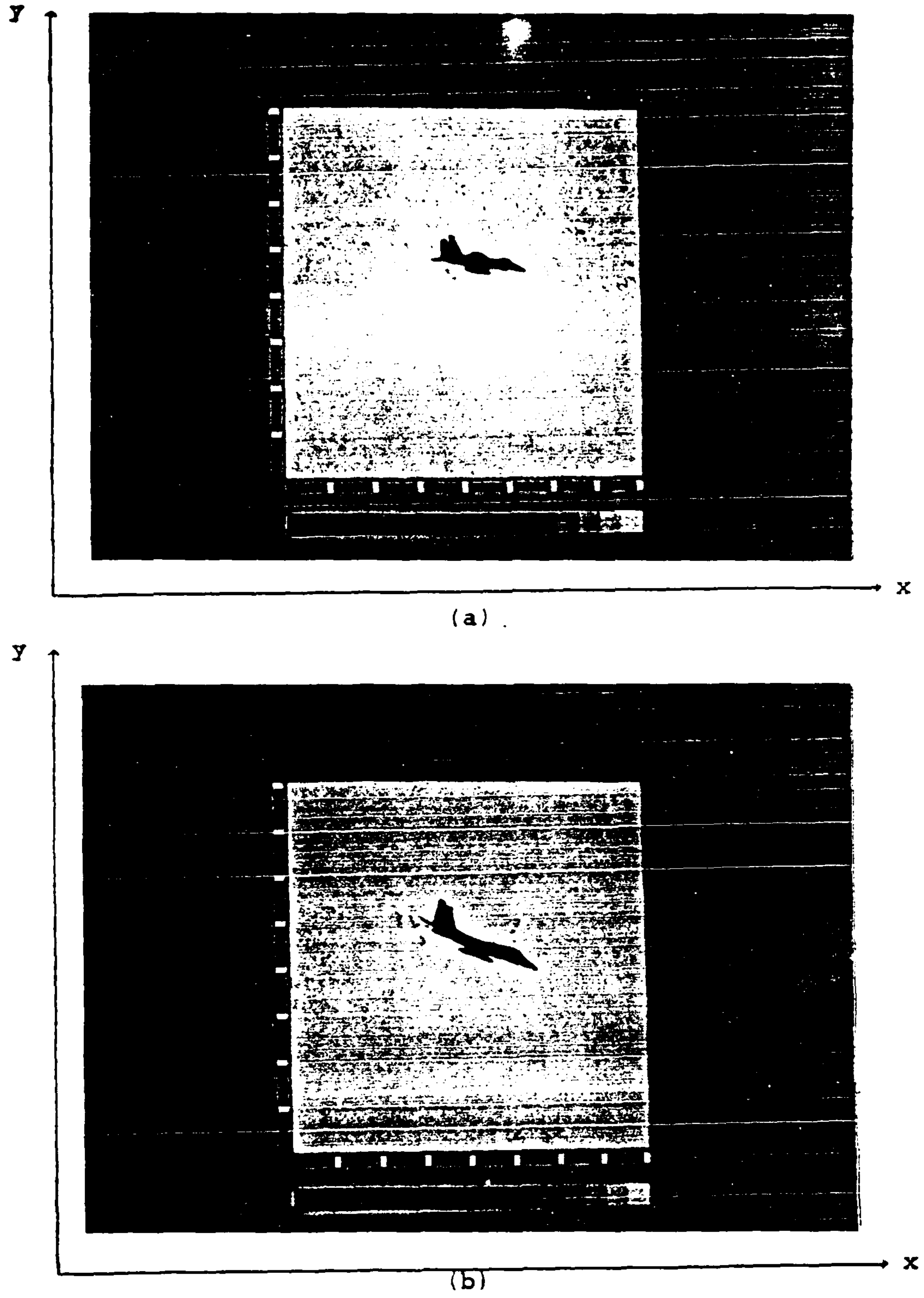
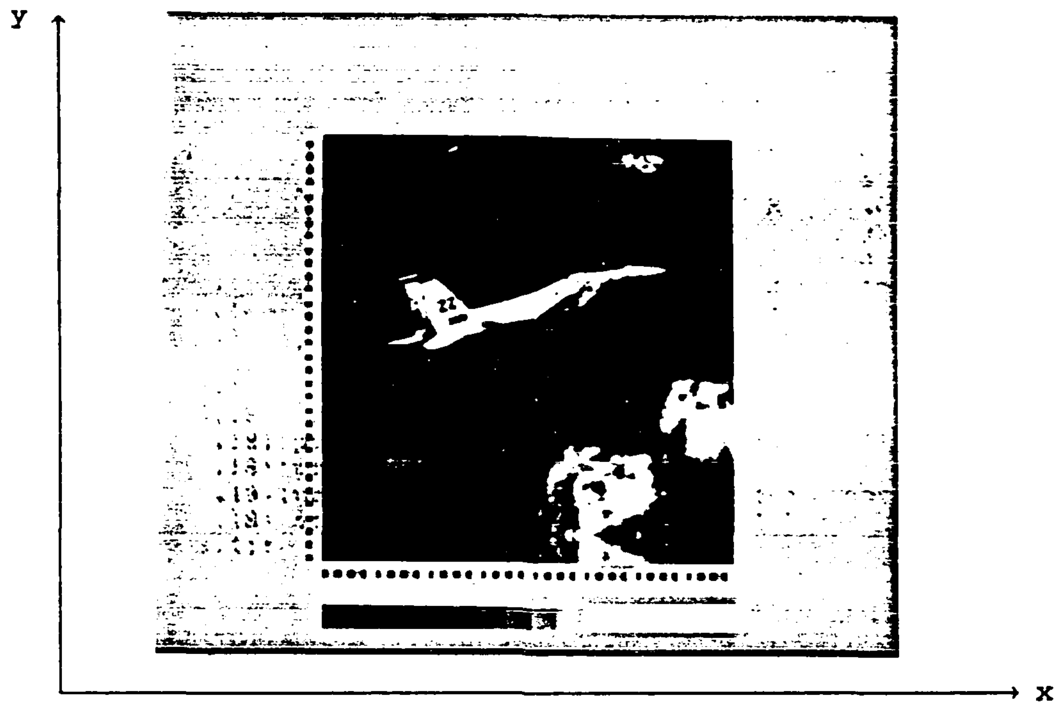


Figure 4.5. (a) Template of F15 (b) Scaled and rotated version of template.



(a)

Figure 4.6. F15 scene.

The ink blot did not act as a pure block of the first few harmonics but seemed to block in some areas and only attenuate others. More importantly, the filter was non-circular which caused the $|FT's|^2$ to be non-symmetric. A very serious problem arose since a characteristic of the FT is there inherent symmetry.

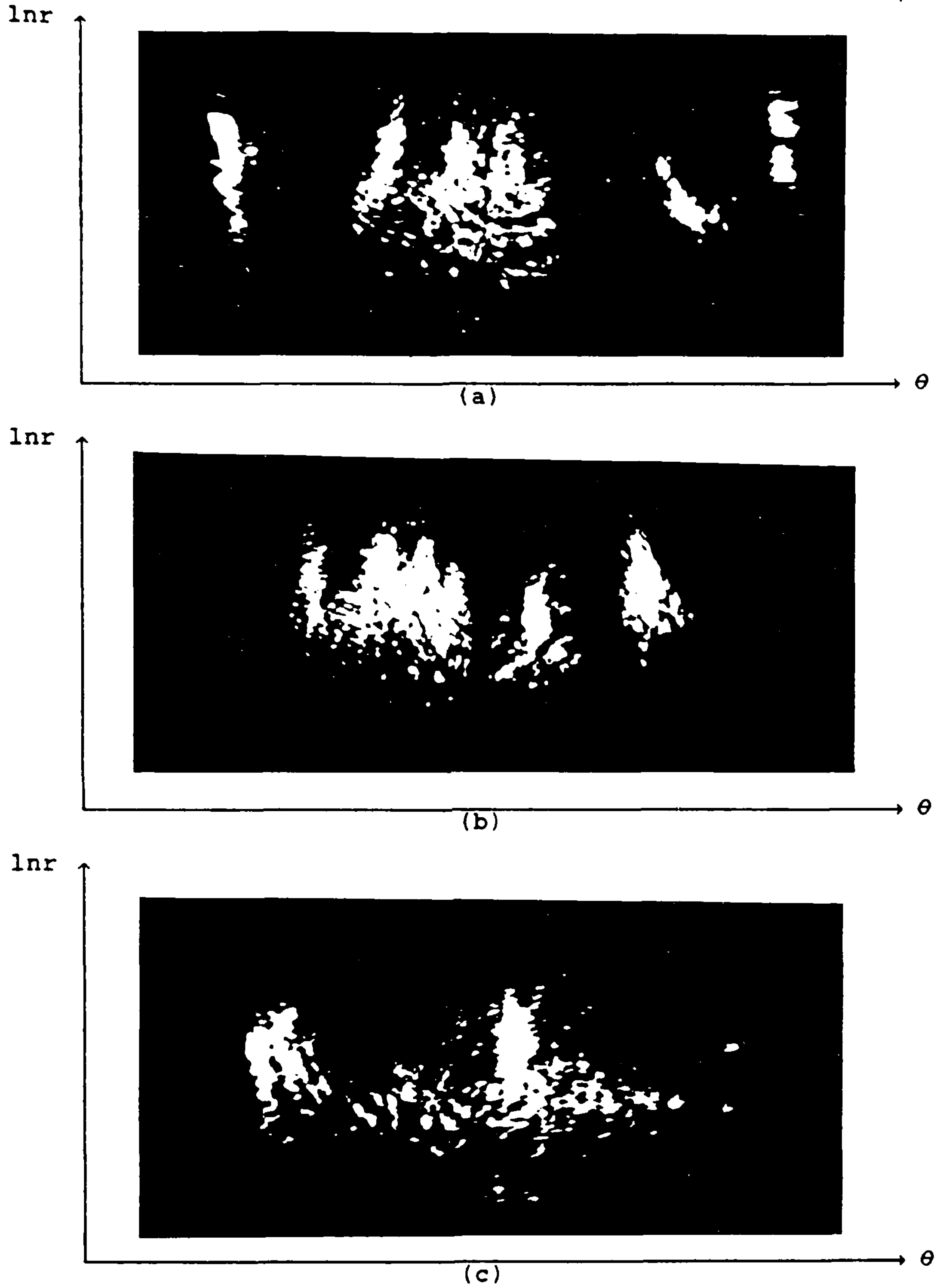


Figure 4.7. PSRI feature spaces. (a) F15 template (b) Scaled and rotated F15 (c) real scene.

The last and most troublesome problem was in the centering of the $|FT|^2$ slides onto the CGH. For the lnr-$\theta$ CT to exist in the diffracted order of plane $P_4$, the input object absolutely <u>must</u> be centered. Since each $|FT|^2$ was mounted slightly different in its respective slide, they all had to be individually centered on the CGH. That was a very subjective process when the PSRI feature space result was unknown. That extra variable led to huge errors when off center by only a few tens of microns. This would not have been a problem if the SLM was used since each $|FT|^2$ would have been detected in the same location and then transferred to the SLM and displayed in precisely the same location each time. Therefore, once the system was centered it would not have to be adjusted again.

## Conclusion

This chapter developed the PSRI feature space using a FT and a lnr-$\theta$ CT CGH in an optical processor. The system was shown to work in real time using a magneto-optic spatial light modulator. The next chapter will give background theory on an optical correlator using a matched spatial filter. Also included in the next chapter will be a description of how the PSRI feature space could be used as the input to the optical correlator built for this thesis effort.

# V. Optical Correlation

## Introduction

Optical correlation has been mentioned throughout this thesis as a method to "search" an input scene for a particular reference target. The problems with the correlator which arise from scale and rotation changes in the input with respect to the reference (template) were used as the basis to transform the input scene into a PSRI feature space.

The PSRI feature space of the reference target (template) would be used to construct a matched spatial filter (MSF) and placed into the correlation plane of the optical correlator. The PSRI feature space of a scene would then be used as the input to the optical correlator. The shape and intensity of the resultant correlation peak (spike) would disclose whether the reference target was located in the scene.

Although no MSF's were constructed and no correlation experiments performed, the inclusion of this chapter is for completeness of the optical pattern recognition system. The first section presents the theory of MSF's and optical correlators using the method of A. Vander Lugt (10:171-192). The next section explains the experimental configuration that was constructed in this thesis effort. Again, no experimental results using the optical correlator were obtained due to a time limitation.

**Theory**

A simplified optical correlator is shown in Figure 5.1. An input, $g(x_1,y_1)$, is placed in the front focal plane ($P_1$) of lens $L_1$. Lens $L_1$ takes the FT of the input producing $G(f_x,f_y)$ at plane $P_2$. A mask of transmittance $H(f_x,f_y)$ is placed in plane $P_2$ to function as the MSF (10:178). The product of G and H exists immediately after plane $P_2$ and lens $L_2$ takes the FT of this product and produces $F\{GH\}$ in plane $P_3$.

The correlation between two functions, g and h, can be found by taking the FT of the product $GH^*$ where $G=F\{g\}$ and $H=F\{h\}$ (10:175). By creating a mask of intensity $H^*$ (complex conjugate of H) in plane $P_2$ of Figure 5.1, the optical processor is performing the correlation between the input g and a reference function h. If the input function is identical to the reference function, then the output will

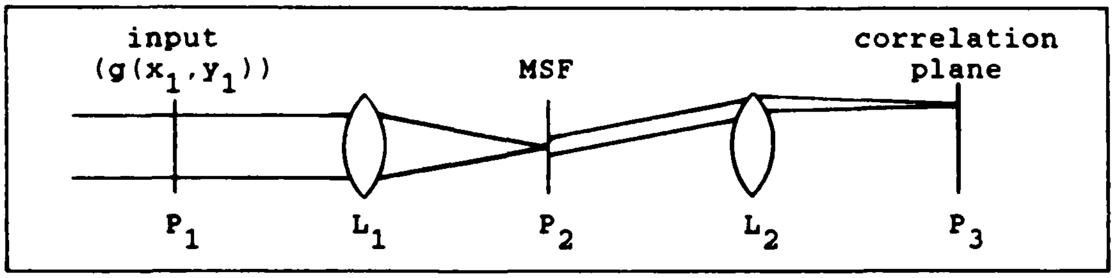


Figure 5.1. Optical correlator using MSF.

be the auto correlation, and an intense spike will appear in plane $P_3$. Otherwise, a cross correlation will be performed resulting in a spike in plane $P_3$ whose intensity is dependent on the similarity between the input and reference.

The MSF can be created using a Vander Lugt filter which incorporates an interferometric system as shown in Figure 5.2. The desired impulse response, reference function (h), is placed in plane $P_1$ and Fourier transformed by lens $L_2$. The FT of h is interfered with a plane wave incident on the film in plane $P_2$ at an angle $\theta$.

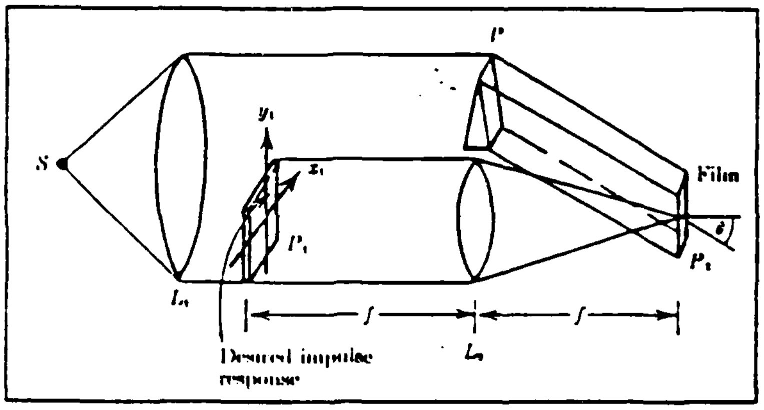


Figure 5.2. MSF construction using Vander Lugt filter (10:172).

The total intensity distribution on the film is determined by the two amplitude distributions present, yielding (10:172):

$$I(x_2,y_2) = \left| r_o \exp(-j2\pi\alpha y_2) + \frac{1}{\lambda f} H\left(\frac{x_2}{\lambda f},\frac{y_2}{\lambda f}\right) \right|^2$$

$$= r_o^2 + \frac{1}{\lambda^2 f^2}\left|H\left(\frac{x_2}{\lambda f},\frac{y_2}{\lambda f}\right)\right|^2 + \frac{r_o}{\lambda f} H\left(\frac{x_2}{\lambda f},\frac{y_2}{\lambda f}\right)\exp(j2\pi\alpha y_2)$$

$$+ \frac{r_o}{\lambda f} H^*\left(\frac{x_2}{\lambda f},\frac{y_2}{\lambda f}\right)\exp(-j2\pi\alpha y_2) \qquad (14)$$

where $\alpha = \frac{\sin\theta}{\lambda}$ , $r_o$ is the amplitude of the plane wave, and f is the focal length of lens $L_2$. The exposed film is then developed to produce a transparency with an amplitude transmittance proportional to the intensity distribution incident during exposure (10:172):

$$t(x_2,y_2) \propto r_o^2 + \frac{1}{\lambda^2 f^2}|H|^2 + \frac{r_o}{\lambda f} H \exp(j2\pi\alpha y_2)$$

$$+ \frac{r_o}{\lambda f} H^*\exp(-j2\pi\alpha y_2) \qquad (15)$$

The film plate is then placed into plane $P_2$ of the optical correlator (Figure 5.1) to function as the MSF. With $g(x,y)$ as the input, the field directly behind the mask is (10:175)

$$U_2 \propto \frac{r_o^2 G}{\lambda f} + \frac{r_o}{\lambda^2 f^2}|H|^2 G + \frac{r_o}{\lambda^2 f^2} HG \exp(j2\pi\alpha y_2)$$

$$+ \frac{r_o}{\lambda^2 f^2} H^* G \exp(-j2\pi\alpha y_2) \qquad (16)$$

Lens $L_2$ of Figure 5.1 performs the FT on the field described in Equation (16) and produces (10:175)

$$U_3(x_3,y_3) \propto r_o^2 g(x_3,y_3) + \frac{1}{\lambda^2 f^2}[h(x_3,y_3)*h^*(-x_3,-y_3)*g(x_3,y_3)]$$
$$+ \frac{r_o}{\lambda f}[h(x_3,y_3)*g(x_3,y_3)*\delta(x_3,y_3+\alpha\lambda f)]$$
$$+ \frac{r_o}{\lambda f}[h^*(-x_3,-y_3)*g(x_3,y_3)*\delta(x_3,y_3-\alpha\lambda f)] \qquad (17)$$

in plane $P_3$. The fourth term of this equation is the cross correlation of g and h centered at $(0,\alpha\lambda f)$ and the one of concern in an optical correlator. Certain procedures must be followed when constructing the filter to assure that the other terms in the output do not overlap with the correlation term (10:175).

## Experimental Configuration for Optical Correlator

The experimental configuration for optical correlation using a MSF is shown in Figure 5.3. The correlator was built into the existing PSRI transformation processor by adding a beamsplitter ($BS_1$) to create a reference beam. The PSRI feature space would be place in the input plane $P_1$ and a MSF would be either created or placed in plane $P_5$ by interfering the FT of the PSRI feature space with the reference beam. To perform an optical correlation, the exposed MSF would be placed in plane $P_5$ and the reference beam would be blocked. The correlation of the input PSRI feature space with the MSF (of a template) would appear in plane $P_6$.

A similar configuration was used by David Neidig (15) to make MSF's and perform optical correlations. MSF's could be created using the configuration in Figure 5.3 by placing a high resolution film plate in plane $P_5$ and interfering the reference beam with the FT of the input. However, before any optical correlation could be performed, an energy normalization technique would have to be incorporated into the system as described by Neidig (15:12-14).

Additional comments about the optical correlator are presented in the Recommendations section of the next chapter.

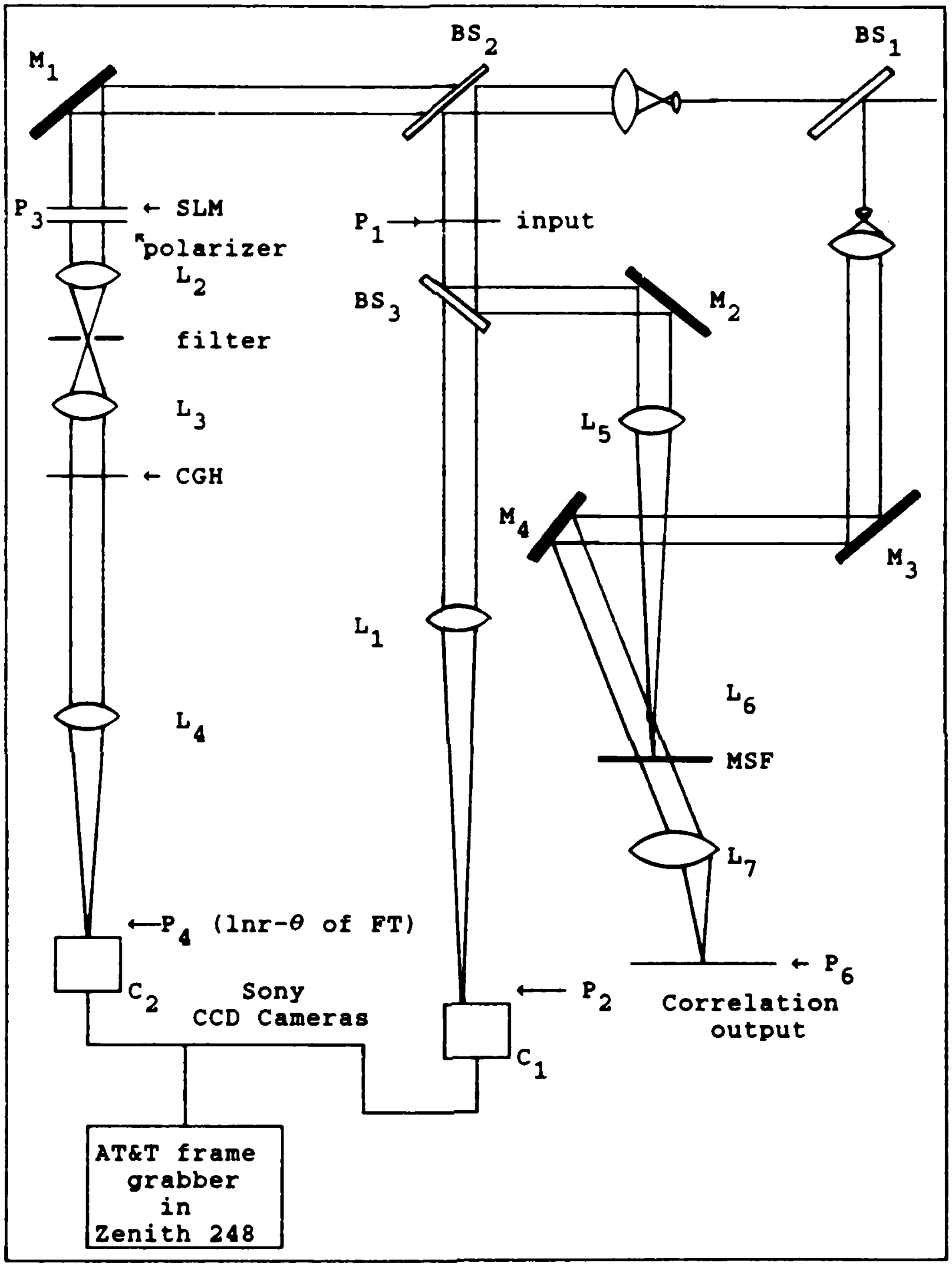


Figure 5.3. PSRI feature space optical processor and optical correlator using MSF's.

## Conclusions and Recommendations

### Conclusions

The Litton Iron Garnet H-Triggered Magneto-Optic Device (LIGHT-MOD) was successfully modified and integrated to function as a spatial light modulator (SLM) in a real-time optical processor. An environment was set-up which integrated the LIGHT-MOD with a Zenith 248 microcomputer and AT&T frame grabber.

Secondly, a real-time rotation and scale (but not position) invariant optical system utilizing a computer generated hologram (CGH) was constructed. The system demonstrated excellent performance against a broad range of non-cluttered objects, targets, and scenes.

A position, scale, and rotation invariant (PSRI) optical system was tested in the final section. Results from this system were inconclusive since correlation was not performed. The magnitude squared of the Fourier transforms ($|FT|^2$) were slightly distorted due to an imperfect high pass filter. After the LIGHT-MOD physically failed, the centering of inputs onto the CGH became a serious problem and added distortions/errors to the final PSRI feature space.

An optical correlator was constructed but never tested.

## Recommendations

For the optical system to function as a real-time PSRI pattern recognition system certain areas must be further investigated.

1) A new magneto-optic substrate should be acquired for the LIGHT-MOD. The LIGHT-MOD should then be used as a SLM to display the $|FT|^2$ of an input onto the CGH. Doing so would circumvent the previously mentioned centering problem and give a real-time capability.

2) A set of various sized high pass filters should be created to improve the $|FT|^2$ and get rid of unwanted harmonics. They can be created using the Dekacon optical system described in Appendix B by imaging circles on high resolution film plates.

3) Another SLM or a liquid crystal display TV could be used to function as a variable high pass filter.

4) The programs written in C to integrate the LIGHT-MOD/Zenith 248/frame grabber system should be added to and improved upon to incorporate more functions of the AT&T frame grabber, increase the display speed of the LIGHT-MOD when used in conjunction with the CCD cameras and frame grabber, and add more functions/options to create a more versatile system.

5) The optical correlator should be fully tested and then used to correlate the PSRI feature spaces. An energy normalization technique should also be incorporated.

6) The PSRI feature spaces obtained using the optical system should be transferred to the VAX and correlated using the Kobel-Martin algorithm (14). Likewise, the PSRI feature spaces obtained with the Kobel-Martin algorithm could be used as inputs to the optical correlator.

# Appendix A

## Software Tools

This Appendix describes and presents the software programs used in conjunction with the thesis effort. The first section describes the vendor provided software: AT&T TARGA software which controls the frame grabber installed on the Zenith 248. Only brief descriptions are provided since a full description can be found in the manual (1). An explanation of the required software to display images on the Litton spatial light modulator (SLM) is presented in the second section. The last section gives a detailed description of the software written to interface the AT&T frame grabber with the Litton SLM.

All software was written in C using the Computer Innovations C86 compiler. In order to modify or add to the existing software, one must become familiar with the C programming language, a C compiler (Computer Innovations C86 is installed on the Zenith 248 which is interfaced with the optical system), and a linker. The programs developed by the researcher and discussed here are included at the end of this Appendix.

### AT&T TARGA System

The AT&T TARGA board is an analog-to-digital converter for a frame of video data which is input as composite video from a charge coupled device (CCD) camera. The TARGA system then digitizes the data on a 0 to 255 grey scale level in a 512x400 pixel array. The image has a higher resolution at the TARGA board since the Sony CCD array is 491x384. These digitized frames can be saved to disk by using the supplied AT&T TARGA software.

The TARGA board is accompanied by two diskettes of software: (1) TARGA software tools and (2) the TARGA demonstration disk. The software tools are a library of C programs that utilize the TARGA board functions. Each of these programs (subroutines) can be compiled individually to be called by other programs or used alone. TESTTARGA and TRUE_ART are the two demonstration programs. TESTTARGA allows the user to run individual subroutines from the tools library to better understand, modify, or debug them. TRUE_ART is a user friendly program that utilizes all of the TARGA tools and allows the user to fully operate the camera and frame grabber. TRUE_ART is a very versatile program which was used greatly during the thesis effort to view optical results, capture and store frames, write text on images, and draw on images.

All of the TARGA programs reside on the hard disk in the directory Targa and TARGA software tools are in the sub-directory Targa\Tools. The program TRUE_ART.EXE is invoked by running TARGA8.BAT in the Targa directory. An AT&T logo appears on the screen and the user is instructed to press the Enter key. A menu is displayed and selections are made by moving the cursor with the up/down cursor keys to the desired function and pressing the F10 key. F10 and F9 are used to invoke all the TARGA commands, not the Enter key. F10 is used to execute a function and F9 is pressed to go back one menu. It should be noted that the cursor can be made to move through the menus faster by pressing the Scroll Lock key. See the TARGA user's manual for a description and operation of all commands and their functions (1).

## LIGHT MOD Software

The first program written for the LIGHT MOD was RASTER.C. This program erases the LIGHT MOD display, writes horizontally to the display (15 rows at a time), erases the display, writes vertically to the display (15 columns at a time), then erases the display again. This sequence is followed 20 times before the program is exited. RASTER.C was used to test and fine tune the LIGHT MOD after the interface and modification was performed. While this program was running, the X and Y current levels, print pulse, and coil pulse were adjusted until each pixel on the display was turned all the way "on" and "off", thus obtaining the greatest dynamic range. A full frame could be displayed in approximately 300msec.

The full display is erased by sending an X-Y current pulse to every pixel followed by a counterclockwise coil pulse. A pixel is accessed (turned on or off) in the software by poking data to a specific offset address. The data represents a row on the SLM and the offset address represents a column (row and column both can take on values from 0 to 127). The Write pulse is activated any time the offset address is less than 2049. Therefore, anytime a column is addressed the Write pulse is generated to select that pixel and place it in its neutral state. Any poke to offset address 2049 will cause the coil to pulse which saturates any pixel in the neutral state and completes the write or erase function.

In C language, the command to poke data to certain addresses is

```
                pokeb(offset,segment,byte)

         where:  offset = the offset segment relative
                          to the segment value.
                segment = segment address of the device
                          (for the LIGHT MOD it is at
                           hex B000).
                   byte = value or data to be sent to
                          the offset.
```

The modification and interface was built in such a way that the offset value directly corresponded to a column and the byte value directly corresoponded to a row. The row had to be logically ORed with hex 80 when a pixel was to be written. This logical OR operation sent SPA7 high and caused the next coil pulse to be in the clockwise direction. When a pixel was to be erased, the actual value of row (0-127) was all that was needed since that defaulted SPA7 to a low. With SPA7 low, the coil pulse would be counterclockwise causing the pixel to turn off. The program RASTER.C shows exactly how to write or erase each and every pixel in the LIGHT MOD display. Any other access to the LIGHT MOD display is a variation of this routine.

<u>Interface</u> <u>Software</u>

Once the LIGHT MOD was shown to be operational, it was necessary to interface the device with the Zenith 248 in order to display real scenes. The remainder of the programs mentioned in this Appendix allow one to manipulate the camera/frame grabber system and display stored or live images on the LIGHT MOD.

TARGASLM.C is the user friendly program which interfaces the AT&T TARGA system frame grabber to the LIGHT MOD. This program initializes the system and presents self explanatory menus which can be chosen to operate the frame grabber and/or display images on the LIGHT MOD. TARGASLM.EXE is located on drive C (the hard disk) under the directory Targa on the Zenith 248 which is interfaced with the optical system. This and all other programs developed for this thesis are also on a 5 1/4 " floppy diskette labeled SLM_PROGRAMS.

The desired function in TARGASLM is executed by typing the associated letter (shown in the left column of the menu) followed by the enter key.

This program calls various AT&T TARGA programs from the

TARGA library as well as two programs to display images on the LIGHT MOD: CCDSLM.C and SLM.C. All TARGA programs are described in the AT&T TARGA manual (1).

CCDSLM.C and SLM.C are very similar programs which are derived from the TARGA program GETPIC. In these, an image file from disk is opened, converted to a 128x128 binary image file, and sent to the LIGHT MOD. Both have a variable thresholding value which is input by the user to convert the 255 grey scale TARGA image to a binary image (each pixel either on or off). The threshold value (0-255) is chosen to obtain the best possible binary image on the LIGHT MOD. CCDSLM.C takes the full 512x400 TARGA image and sends every 4th column and ever 3rd row to the 128x128 LIGHT MOD. This allows the full image to be displayed on the LIGHT MOD but the image becomes distorted and looses image resolution due to the sampling technique.

SLM.C only takes the center 256x256 of the TARGA image and sends every other pixel to the LIGHT MOD in order to obtain higher resolution of the image on the LIGHT MOD and prevent distortion of the image.

All programs are commented well and therefore only described breifly here. Any modifications or additions to TARGASLM or the other programs must be followed by proper compilation and linking. It is very important that all called programs are included during the linking process. After compiling the program (run CC.BAT when using C86), the linker is invoked by running LINK. The compiled programs object file (.OBJ) must be included followed by the .OBJ files of all called programs that are not in a library. The libraries C86S2S and TARGA must also be included since C86S2S contains all the C programming functions and the TARGA library contains the .OBJ files of all the TARGA tools programs. The system compiler and linker are described in the appropriate manuals.

```
/*
 * Program: RASTER.C        Written by: Michael W. Mayo
 *                                      20 July 1987
 *
 *     This program is used to test the operation of the
 * Litton spatial light modulator (SLM).  Rows and columns
 * are alternately turned on (15 at a time) while the display
 * is erased after each horizontal (rows) or vertical (cols)
 * write.
 *     While the program is running, the polarizer which is
 * located after the SLM can be rotated until the desired
 * dynamic range is reached (that is, an 'on' pixel is as
 * bright as it can be and an 'off' pixel is as dark as it
 * can be).  Other adjustments can be made to the X- and
 * Y-current levels, WRITE pulse width, and COIL pulse width.
 * However, one must be very careful when adjusting these
 * parameters.  They are set at what I feel are optimum settings
 * and their adjustment should be avoided if at all possible
 * (refer to the Litton LIGHT MOD manual).
 */

#define ON_MASK 0x80      /*  Logical OR with row sets SPA7 high
                           *  to allow a pixel write
                           */

extern unsigned char pokeb();
unsigned segment=0xb000;     /* Segment address of LIGHT MOD device.  */
unsigned row, col;           /* Row and Col of the LIGHT MOD display. */
char byte;                   /* Data sent to address. */

main()
{
   int loop;
   int a;

          /* Goes through program 20 times. */

   for (loop=0; loop<=20; loop++)   {

   erase( );  /* Calls function to erase the entire display. */

/*
 * Section writes horizontally to the LIGHT MOD
 * 15 rows at a time.
 */
   printf("\n\tBeginning horizontal write to SLM.\n");
   a=0;
   while (a<128){
      for(row=a; row<=a+15; row++){
         for(col=0; col<128; col++){
            byte = row | ON_MASK;      /* Puts SPA7 high for
                                          write operation */

            pokeb(col,segment,byte);  /* PT pulse to nucleate pixels */
            }
         }
      pokeb(2049,segment,255);        /* PC pulse to saturate pixels */
      a+=16;
      }
   printf("\n\tHorizontal write complete.\n");
```

```
    erase();

/*
 * Section writes vertically to the LIGHT MOD
 * 15 columns at a time.
 */
    printf("\n\tBeginning vertical write to SLM.\n");
    a=0;
    while (a<128){
       for(row=0; row<127; row++){
          for(col=a; col<=a+15; col++){
             byte = row | ON_MASK;
             pokeb(col,segment,byte);    /* PT pulse to nucleate pixels */
             }
          }
       pokeb(2049,segment,255);          /* PC pulse to saturate pixels */
       a+=16;
       }
    printf("\n\tVertical write complete.\n");
    }
    printf("\n\nProgram over!");
}

erase()      /* Function erases the entire SLM */
{
    printf("\n\n\tErasing SLM display.\n");
    for(row=0; row<=127; row++)  {
       for(col=0; col<=127; col++)  {
          byte = row;
          pokeb(col,segment,byte);    /* PT pulse to nucleate pixels */
          }
       }
    pokeb(2049,segment,255);          /* PC pulse to saturate pixels */
    printf("\n\tErase operation complete.\n\n");
}
```

```
/* Prgm: TARGASLM.C   Written by Michael W. Mayo, 20 Aug 1987
 * Subroutines: ccdslm, slm
 * TARGA subroutines: GraphInit, GetPic, SetLiveMode,
 *                    GrabFrame,        PutPic, GraphEnd.
 *
 *     This is a user friendly program (I hope!) which
 * interfaces the AT&T TARGA frame grabber system with the
 * Litton spatial light modulator.  From this program you
 * can operate the camera/frame grabber and/or display
 * images on the spatial light modulator (SLM).  More
 * modifications could be added to this program so all
 * the TARGA software functions can be incorporated.  Due
 * to a lack of time and necessity, only the above mentioned
 * TARGA programs were incorporated.  To get full use of the
 * TARGA system, the AT&T provided program TRUE_ART.EXE should
 * be used.
 *
 */

#include "tardev.h"
#include <stdio.h>
extern struct TARstruct *targa;

#define  CLS  "\033[2J"       /* Escape code for clear screen */
#define  CUP  "\033[%d;%dH"   /* Moves cursor to position row, col */

main()
{
setup();
menu();

printf("\033[0m");         /* All attributes off: black on white */
printf(CLS);
}

setup()
{
 printf("\033[44;30m");/* Sets a blue background with black letters */
 printf(CLS);           /*
                         * Clears screen and positions cursor
                         * in upper left corner.
                         */
 printf(CUP,7,20);
 printf("            AT&T FRAME GRABBER and ");printf(CUP,8,20);
 printf("SPATIAL LIGHT MODULATOR OPERATING SYSTEM");
 printf(CUP,21,30); printf("Press ENTER to begin");
 printf("\033[5m");printf(CUP,16,28);printf("\033[36m");
 printf("Turn on High res monitor ");printf(CUP,17,28);
 printf("       and camera         ");
 printf(CUP,23,1);
 while(getchar() != '\n');        /* waiting for return key    */

 printf(CLS);
 printf(CUP,11,28);printf("INITIALIZING TARGA SYSTEM\n");
 if (GraphInit(-1)==-1) {
        printf("\t\t\t\n RASTER COUNTER IS BROKEN\n");
        printf("\t\t\t I/O SPACE PROBABLY WRONG\n\n");
        printf("\t\t\t GRAPHINIT FAILED");
        }
        printf("\033[0m");           /* All screen attributes off */
}
```

```
menu()    /* Sets up screen for main menu */
{
 printf("\033[41m");printf(CLS);  /* Red background; clear screen */
 printf("\033[44;32m");           /* Black window, green letters */

 printf(CUP, 3,18); printf("                    MAIN MENU                   ");
 printf(CUP, 4,18); printf("                                                ");
 printf(CUP, 5,18); printf("     key                   function             ");
 printf(CUP, 6,18); printf(" =========  ============================        ");
 printf(CUP, 7,18); printf("     L      Enter LIVE camera mode.             ");
 printf(CUP, 8,18); printf("     T      Display TARGA image on monitor      ");
 printf(CUP, 9,18); printf("              and spatial light modulator       ");
 printf(CUP, 10,18);printf("              (.tga file from disk).            ");
 printf(CUP, 11,18);printf("              Full 512x400 .tga image is        ");
 printf(CUP, 12,18);printf("              displayed on 128x128 SLM.         ");
 printf(CUP, 13,18);printf("     D      DISPLAY only center 256x256         ");
 printf(CUP, 14,18);printf("              of the image on the 128x128 SLM");
 printf(CUP, 15,18);printf("                                                ");
 printf(CUP, 16,18);printf("                                                ");
 printf(CUP, 17,18);printf("     Q      Exit program (QUIT).                ");
 printf(CUP, 18,18);printf("                                                ");
 printf(CUP, 19,18);printf(" --press key for desired function               ");
 printf(CUP, 20,18);printf("   followed by the ENTER key.                   ");
 printf(CUP, 23,1);

 keysense();
}

keysense()
{
   int key;
   int Found;
   int i;
   char infile[50];

   GraphInit(-1);
   key = getchar();
   switch (key)
     {
     case 'l':
     case 'L':
       Found = 2;
       SetLiveMode();
       break;
     case 't':
     case 'T':
       Found = 1;
       printf("\033[40H");printf(CLS);printf("\033[32H");
       printf(CUP,11,20);printf("This section will take a .TGA image file and"
       printf(CUP,12,20);printf("display the binary version on the Litton SLM."
       printf(CUP,15,20);printf("Please input the TARGA file -> ");
       scanf("%s\n",infile);
            printf(CUP,20,24);printf("DISPLAYING IMAGE ON MONITOR");
       GetPic(infile,-1,-1,-1);

       printf(CLS);
       ccdslm(infile,-1,-1,-1);
       break;
     case 'd':
     case 'D':
       Found = 1;
       printf("\033[40H");printf(CLS);printf("\033[32H");
       printf(CUP,11,20);printf("Only center 256x256 of a .TGA image file will"
       printf(CUP,12,20);printf("be displayed on the 128x128 Litton SLM.");
```

```
          printf(CUP,15,20);printf("Please input the TARGA file -> ");
          scanf("%s\n",infile);

          printf(CUP,20,24);printf("DISPLAYING IMAGE ON MONITOR");
          GetPic(infile,-1,-1,-1);

          printf(CLS);
          ccdslm(infile,-1,-1,-1);

          slm();
          break;
        case 'q':
        case 'Q':
          GraphEnd();
          printf("\033[0m");
          printf(CLS);
          exit();
          break;
        default :
          Found = 0;
          printf("\033[44;30m");
          printf(CUP, 23,1);printf("\033[s");printf("  KEY NOT ASSIGNED");
          for (i=0; i<=32000; i++)
            {
          /* pause to display message */
            }
          printf("\033[u");printf("\033[K"); /* Erase line */
          break;
      }
    if (Found == 0) keysense();    /*
                                    * Invalid input. Get another key.
                                    */
    if (Found == 1) menu();
    if (Found == 2) LiveMode();
}

LiveMode()
{
 printf("\033[41m");printf(CLS);  /* Red background; clear screen */
 printf("\033[44;32m");           /* Black window, green letters */

 printf(CUP, 3,18); printf("                LIVE CAMERA MODE              ");
 printf(CUP, 4,18); printf("                                              ");
 printf(CUP, 5,18); printf("   key             function                   ");
 printf(CUP, 6,18); printf(" =========  ============================       ");
 printf(CUP, 7,18); printf("     C      CAPTURE a video frame in the      ");
 printf(CUP, 8,18); printf("                    display memory.           ");
 printf(CUP, 9,18); printf("     S      STORE image displayed on          ");
 printf(CUP, 10,18);printf("                 monitor in a disk file.      ");
 printf(CUP, 11,18);printf("                                              ");
 printf(CUP, 12,18);printf("                                              ");
 printf(CUP, 13,18);printf("                                              ");
 printf(CUP, 14,18);printf("                                              ");
 printf(CUP, 15,18);printf("                                              ");
 printf(CUP, 16,18);printf("     M      Return to MAIN menu.              ");
 printf(CUP, 17,18);printf("     Q      Exit program (QUIT).              ");
 printf(CUP, 18,18);printf("                                              ");
 printf(CUP, 19,18);printf(" --press key for desired function             ");
 printf(CUP, 20,18);printf("   followed by the ENTER key.                 ");
 printf(CUP, 23,1);
 Live();
}
```

```
Live()
{
   int key;
   int Found;
   int i;
   char infile[50];

   key = getchar();
   switch (key)
      {
      case 'c':
      case 'C':
         Found = 2;
         GrabFrame();
         GraphInit(-1);
         break;
      case 's':
      case 'S':
         Found = 2;
         printf("\033[40H");printf(CLS);printf("\033[32H");
         printf(CUP,12,20);printf("Input filename -> ");
         scanf(" %s\n",infile);
         printf(CUP,15,20);printf("Saving image to disk as a .TGA file.");
         PutPic(infile,-1,-1,-1);
         printf(CUP,18,20);printf("Image saved.  Hit ENTER to continue.");
         getchar();
         getchar();
         break;
      case 'm':
      case 'M':
         Found = 1;
         break;
      case 'q':
      case 'Q':
         GraphEnd();
         printf("\033[0m");
         printf(CLS);
         exit();
         break;
      default :
         Found = 0;
         printf("\033[44;30m");
         printf(CUP, 23,1);printf("\033[s");printf("  KEY NOT ASSIGNED");
         for (i=0; i<=32000; i++)
            {
         /* pause to display message */
            }
         printf("\033[u");printf("\033[K"); /* Erase line */
         break;
      }
   if (Found == 0) Live();            /*
                                       * Invalid input. Get another key.
                                       */
   if (Found == 1) menu();
   if (Found == 2) LiveMode();
}
```

```
/* Prgm: CCDSLM.C          Written by Michael W. Mayo, 10 Aug 1987
 * Calling Program: TARGASLM.C
 *
 * This function is called by TARGASLM to display a TARGA image
 * (.TGA) onto the Litton spatial light modulator.  The .TGA image
 * is a 512x400 resolution image; each pixel having an 8 bit gray
 * scale value between 0 and 255.  This function takes the 512x400
 * image and compresses it to a 128x128 binary image (a pixel on
 * on the SLM is either on or off - no gray levels).
 *
 * CCDSLM.C is a modified version of the TARGA library function
 * GETPIC.C.  Instead of opening the .TGA disk file and dumping
 * it to the high resolution monitor, the file is intercepted,
 * compressed, binarized, and sent to the SLM.
 *
*/


#include <stdio.h>
#include  tardev.h"

#define  MAXBUFSIZE  32767  /*   must be at least 2048 */
#define  ON_MASK  0x80      /* Allows write to SLM by putting SPA7 high */
#define  OFF_MASK 0x7f

extern  char  *malloc();

extern  struct TARStruct *targa;
static  struct hdStruct header;
extern  char   *strchr();

extern unsigned char pokeb();
unsigned segment=0xb000;        /* Offset segment of poke address */
unsigned slmrow, slmcol;        /* Row and col of SLM (0-127)     */
char byte;                      /* Byte data sent to poke address */

int ccdslm(picture,XCorner, YCorner, screen)
char    picture[];
int     XCorner, YCorner;
int screen;
{
char *buffer;
char *ColorMap;
char *dataPntr;
int     pict;
unsigned buffSize;

char inname[50];
int points;
int xOffset, yOffset;
int RowsPerRead, BytesPerRead, BytesInPixel;
int BytesInRow;
int row, i, col;
int mapBytes;
int mapLength;
int Convert;        /*  flag to indicate type of convertion required */
int Compressed;     /*  flag to indicate if Compressed image */
int InputBytes;     /*  number o bytes per input pixel */
int PixPerRead;     /*  number of pixels in one read */
int intensity;      /*  0-255 intensity value of a pixel */
int threshold;      /*  Threshold value for binary write to SLM */
```

```
/*
 *  GET TARGA  FILE
 *
 *  Compile the name first
 */

    strcpy(inname,picture);
    if (strchr(inname, '.')==NULL) strcat(inname,".tga\0");

/*
 * open the file header and see if a valid targa image
 */

    if(GetPicHeader(inname,&pict,&header,0) == -1)  {
          printf("\n\r IMAGE %s NOT FOUND\n\r",picture);
          return(-1);
          }

    BytesInPixel  =  targa->BytesPerPixel;     /* bytes/pixel in display */
    if (BytesInPixel == 3 ) BytesInPixel =4;

    InputBytes= (header.dataBits+7)/8;   /*  byte/pixel on the disk */

/*
 *  DEFINE IMAGE ORIGIN
 *
 *  set the corner unless specifiec otherwise
 */
    if ( XCorner == -1 )
       {xOffset = header.XOffset; yOffset = header.YOffset; }
    else
       { xOffset = XCorner; yOffset = YCorner; }

    Convert= 0;        /*  default flag no conversion necessary */
                       /* if color mapped covert the map as needed */

    if ( BytesInPixel != InputBytes  )   Convert = 1;
/*
 * See if compressed file or not
 */
    if ( header.dataType > 8 )   Compressed = 1;
    else                          Compressed = 0;
/*
 *  SET PARAMETERS TO ALLOW MY DISK READ
 */
    points = header.x;
/*
 *  Compute the maximum number of rows that we
 *  can read in at a shot and the number of bytes
 *  to read from the disk (assume uncompressed).
 */
    if ( BytesInPixel > InputBytes )  {BytesInRow = BytesInPixel*points;}
    else                              {BytesInRow = InputBytes*points;}

    if  (Compressed == 0 ) {
       buffSize =  MAXBUFSIZE;
       while ( ( ( (int)(buffer=malloc(buffSize)) ) == NULL) &&
             (buffSize>=BytesInRow)) {
             buffSize = (buffSize+1)>>1;
       }
       if ( buffSize  < BytesInRow ) {
          close(pict);
          return(-2);
          }
```

```
     } else   {
        buffSize = BytesInRow;
        buffer = malloc( BytesInRow);
        if ( ExpStart(pict,InputBytes) < 0 ) {
           free(buffer);
           close(pict);
           return(-2);
           }
     }

     RowsPerRead =  buffSize/BytesInRow;
     BytesPerRead= RowsPerRead * InputBytes * points;
     PixPerRead  = RowsPerRead *points;
/*
 *  Erase the SLM display
 */
erase();
/*
 *  Targa image sent to spatial light modulator.
 */
printf("\033[15;22H");
printf("Input image intensity threshold value: ");
scanf(" %d \n", &threshold);  printf("\033[2J");
printf("\033[10;26H");
printf("Transferring image to the SLM.");
slmrow = 0;  /* First row on SLM */
for ( row=0; row< header.y; row += RowsPerRead )
       {
/*
 *  Read the date from the TARGA file
 */
       if (Compressed == 0) read(pict,buffer,BytesPerRead);
       else ExpRead(buffer,PixPerRead);

       if (Convert == 1)
          UnPackBuf(buffer,ColorMap,PixPerRead,BytesInPixel);

       dataPntr = buffer;
       for (i=0; (i<RowsPerRead)&&((i+row)<=(header.y-8)); i++)
          {
/*
 *  Convert TARGA image from 0-255 gray levels to binaray and
 *  place binary image on SLM.
 */
          if ((row+i)<=7) dataPntr+=512;  /*
                                           * Skip 1st 8 rows but
                                           * update pointer.
                                           */
          else
             {
             if (((row+i+1)%3)==0)   /* Take every 3rd row. */
                {
                slmcol=0;
                for (col=0; col<=511; col++)
                   {
                   if (((col)%4)==0)       /* Every 4th column */
                      {
      /*
       *  Gray levels are at -128 to 127.  Convert so they are 0 to 255
       */
                      intensity=*dataPntr;
                      if (intensity<0) intensity+=256;
                      if (intensity>threshold)
                         {
                         byte = slmrow | ON_MASK;      /* Assures SPA7 is high */
                         pokeb(slmcol,segment,byte);  /* Write pixel to SLM */
```

```
                              }

                           slmcol++;    /* Next col on SLM */
                           }
                        dataPntr++;     /* Update pointer  */
                        }
                     slmrow++;          /* Next row on SLM */
                     }
                  else
                     dataPntr += 512;  /* Skip row but update pointer */
                  }
               }
            }
   pokeb(2049,segment,255);   /* Coil pulse to write image on SLM */
   printf("\033[19;27H");
   printf("Image dump to SLM complete.");
   printf("\033[24;23H");
   printf("Press ENTER to return to Main Menu");
   getchar();   /* Waiting for ENTER key */
   getchar();
/*
 * Write to SLM complete.
 */
   if ( Compressed == 1 ) ExpStop();
   free(buffer);
   close(pict);
   return(0);


/*
 *  This function is used to clear (erase) the spatial
 *  light modulator display.
 */

erase()
{
   int row, col;

   printf("\033[5;30H");                  /* Places cursor at */
   printf("Erasing SLM display.");        /* row 5 and col 30 */
   for (row=0; row<=127; row++)
      {
      for (col=0; col<=127; col++)
         {
         byte = row;
         pokeb(col,segment,byte);         /* Nucleate each row      */
         }                                /* and column on display  */
      }
   pokeb(2049,segment,255);               /* Saturate (coil pulse) to
                                             complete erase         */
   printf("\033[8;28H");
   printf("Erase operation complete.");
}
```

```
/* Prgm: SLM.C          Written by Michael W. Mayo, 5 Sep 1987
 * Called by TARGASLM.C
 *
 *      This program is very similar to CCDSLM.C.  A .TGA image
 * file is retreived from disk, compressed, binarized, and sent
 * to the spatial light modulator.  What makes this function
 * different is that only the middle 256x256 of the 512x400 image
 * is used (in other words, the outside of the image is cut off).
 * The remaining 256x256 image is then sampled (every other pixel)
 * and displayed as a 128x128 image on the spatial light modulator.
 *     This program is needed to increase the image resolution on
 * the spatial light modulator and decrease the sampling error.
 * SLM.C was used mainly to display Fourier transforms of images on
 * the SLM.
*/


#include <stdio.h>
#include "tardev.h"

#define  MAXBUFSIZE  32767  /*   must be at least 2048 */
#define  ON_MASK  0x80      /* Allows write to SLM by putting SPA7 high */
#define  OFF_MASK 0x7f

extern  char  *malloc();

extern  struct TARStruct *targa;
static  struct hdStruct header;
extern  char   *strchr();

extern unsigned char pokeb();
unsigned segment=0xb000;         /* Offset segment of poke address */
unsigned slmrow, slmcol;         /* Row and col of SLM (0-127)     */
char byte;                       /* Byte data sent to poke address */

int slm(picture,XCorner, YCorner, screen)
char     picture[];
int      XCorner, YCorner;
int screen;
{
char *buffer;
char *ColorMap;
char *dataPntr;
int      pict;
unsigned buffSize;


char inname[50];
int points;
int xOffset, yOffset;
int RowsPerRead, BytesPerRead, BytesInPixel;
int BytesInRow;
int row, i, col;
int mapBytes;
int mapLength;
int Convert;        /*  flag to indicate type of convertion required */
int Compressed;     /*  flag to indicate if Compressed image */
int InputBytes;     /*  number o bytes per input pixel */
int PixPerRead;     /*  number of pixels in one read */
int intensity;      /*  0-255 intensity value of a pixel */
int threshold;      /*  Threshold value for binary write to SLM */
```

```
/*
 *       GET TARGA  FILE
 *
 *       Compile the name first
 */

   strcpy(inname,picture);
   if (strchr(inname, '.')==NULL) strcat(inname,".tga\0");

/*
 *  open the file header and see if a valid targa image
 */

   if(GetPicHeader(inname,&pict,&header,0) == -1)  {
         printf("\n\r IMAGE %s NOT FOUND\n\r",picture);
         return(-1);
         }

   BytesInPixel  =  targa->BytesPerPixel;   /* bytes/pixel in display */
   if (BytesInPixel == 3 ) BytesInPixel =4;

   InputBytes= (header.dataBits+7)/8;   /*  byte/pixel on the disk */

/*
 *  DEFINE IMAGE ORIGIN
 *
 *  set the corner unless specified otherwise
 */
   if ( XCorner == -1 )
      {xOffset = header.XOffset; yOffset = header.YOffset; }
   else
      { xOffset = XCorner; yOffset = YCorner; }

   Convert= 0;        /*  default flag no conversion necessary */
                      /* if color mapped covert the map as needed */

   if ( BytesInPixel != InputBytes)     Convert = 1;
/*
 *  See if compressed file or not
 */
   if ( header.dataType > 8 )   Compressed = 1;
   else                          Compressed = 0;
/*
 * SET PARAMETERS TO ALLOW MY DISK READ
 */
   points = header.x;
/*
 *  Compute the maximum number of rows that we
 *  can read in at a shot and the number of bytes
 *  to read from the disk (assume uncompressed).
 */
   if (BytesInPixel > InputBytes)  {BytesInRow = BytesInPixel*points;}
   else                            {BytesInRow = InputBytes*points;}

   if  (Compressed == 0 ) {
      buffSize =  MAXBUFSIZE;
      while ( ( ( (int)(buffer=malloc(buffSize)) ) == NULL) &&
            (buffSize>=BytesInRow)) {
            buffSize = (buffSize+1)>>1;

      }
      if ( buffSize  < BytesInRow ) {
         close(pict);
         return(-2);
         }
   }  else   {
      buffSize = BytesInRow;
      buffer = malloc( BytesInRow);
      if ( ExpStart(pict,InputBytes) < 0 ) {
         free(buffer);
         close(pict);
         return(-2);
         }
   }

   RowsPerRead =  buffSize/BytesInRow;
   BytesPerRead= RowsPerRead * InputBytes * points;
   PixPerRead  = RowsPerRead *points;
/*
 *  Erase the SLM display
 */
erase();
/*
 *  Targa image sent to spatial light modulator.
 */
printf("\033[15;22H");
printf("Input image intensity threshold value: ");
scanf(" %d \n", &threshold);  printf("\033[2J");
printf("\033[10;26H");
printf("Transferring image to the SLM.");
slmrow = 0;  /* First row on SLM */
for ( row=0; row< header.y; row += RowsPerRead )
      {
/*
 *  Read the date from the TARGA file
 */
      if (Compressed == 0) read(pict,buffer,BytesPerRead);
      else ExpRead(buffer,PixPerRead);

      if (Convert == 1)
         UnPackBuf(buffer,ColorMap,PixPerRead,BytesInPixel);

      dataPntr = buffer;
      for ((i=0; ((i<RowsPerRead)&&((i+row)<=(header.y-72)); i++)
         {
/*
 *  Convert TARGA image from 0-255 to binaray and
 *  place binary image on SLM
 */
         if ((row+i)<=71) dataPntr+=512;  /*
                                           * Skip 1st 72 rows but
                                           * update pointer.
                                           */
         else
            {
            if (((row+i)%2)==0)   /* Take every other row. */
               {
               slmcol=0;
               dataPntr+=128;     /* Skip first 128 cols */
               for (col=0; col<=255; col++)
                  {
                  if (((col)%2)==0)       /* Every other column */
                     {
                     intensity=*dataPntr;
                     if (intensity<0) intensity+=256;
                     if (intensity>threshold)
                        {
                        byte = slmrow | ON_MASK;
                        pokeb(slmcol.segment,byte);  /* Write pixel to SLM */
                        }
```

```
                                    slmcol++;
                                    }
                                dataPntr++;      /* Update pointer */
                                }
                            dataPntr+=128;      /* Skip last 128 columns */
                            slmrow++;
                            }
                        else
                            dataPntr += 512;  /* Skip row but update pointer */
                        }
                    }
                }
        pokeb(2049,segment,255);    /* Coil pulse to write image on SLM */
        printf("\033[19;27H");
        printf("Image dump to SLM complete.");
        printf("\033[24;23H");
        printf("Press ENTER to return to Main Menu");
        getchar();    /* Waiting for ENTER key */
        getchar();
    /*
     * Write to SLM complete.
     */
        if ( Compressed == 1 ) ExpStop();
        free(buffer);
        close(pict);
        return(0);
    }

    erase()    /* Erases SLM display */
    {
        int row, col;

        printf("\033[5;30H");
        printf("Erasing SLM display.");
        for (row=0; row<=127; row++)
            {
            for (col=0; col<=127; col++)
                {
                byte = row;
                pokeb(col,segment,byte);
                }
            }
        pokeb(2049,segment,255);
        printf("\033[8;28H");
        printf("Erase operation complete.");
    }
```

## Appendix B

## Fabrication of the GGH

This Appendix describes the proceedure of producing and printing the grating pattern which will perform the desired optical coordinate transformation (CT). Also included is the method used to reduce and place the grating pattern onto a high resolution film plate.

### Computer Generated Inteferogram

The computer generated inteferogram is made up of a binary fringe pattern printed on a laser printer and then photoreduced. Due to the binary nature of the inteferogram, an implicit carrier frequency is inherent in the hologram. As a result, many diffracted waves will result when the CGH is illuminated. A sufficiently high carrier frequency must be chosen to seperate the first-order diffracted wave (where the CT is located) from the higher-order waves. To avoid aliasing (overlapping of diffracted orders), the following equation must be satisfied for the carrier frequency $\alpha$ (6: 939)

$$\alpha > \left[ \frac{3}{\lambda f_L} \right] \ln \left[ \left( \frac{1}{2} x_{max} \right)^2 + \left( \frac{1}{2} y_{max} \right)^2 \right]^{1/2} \qquad \text{(B.1)}$$

where: $x_{max}$ and $y_{max}$ are input dimensions.
$\lambda$ is the wavelength of light used.
$f_L$ is the focal length of CT lens.

The CGH constructed for this thesis effort had the following parameters: $x_{max} = y_{max} = 10mm$ (this is the dimensions of the spatial light modulator); $\lambda = 0.6328\mu m$; and $f_L = 300mm$. Using these parameters in Equation B.1 and solving results in an $\alpha > 30.9$ lines/mm. An $\alpha$ of 35 lines/mm was chosen to meet the above criteria.

To form the computer generated hologram (interferogram), the binary fringes of the transmittance function were printed on a laser printer and then optically reduced. For the lnr-$\theta$ CT CGH the transmittance function obtained by interfering the required phase function given by

$$\phi(x,y) = \left[\frac{2\pi}{\lambda f_L}\right] * \left[\ln(x^2 + y^2)^{1/2} - y\tan(y/x) - x\right] \qquad (8)$$

with a carrier, $\alpha$, is given by

$$t(x,y) = 0.5\left[1 + \cos[2\pi\alpha x - \phi(x,y)]\right] \qquad (B.2)$$

The transmittance function has fringes (maxima) at positions (x,y) satisfying

$$2\pi\alpha x - \phi(x,y) = 2\pi n_k \qquad (B.3)$$

for various values of $n_k$, where each value of $n_k$ corresponds to a different fringe.

The various (x,y) points that satisfy (B.3) for each value of $n_k$ were connected to form the fringe pattern. The Metalib program (for contours) on the AFIT computer systems was used to solve for the x,y points and the resulting fringe pattern was printed on the Imagen laser printer. The CGH plot is shown in Figure B.1.

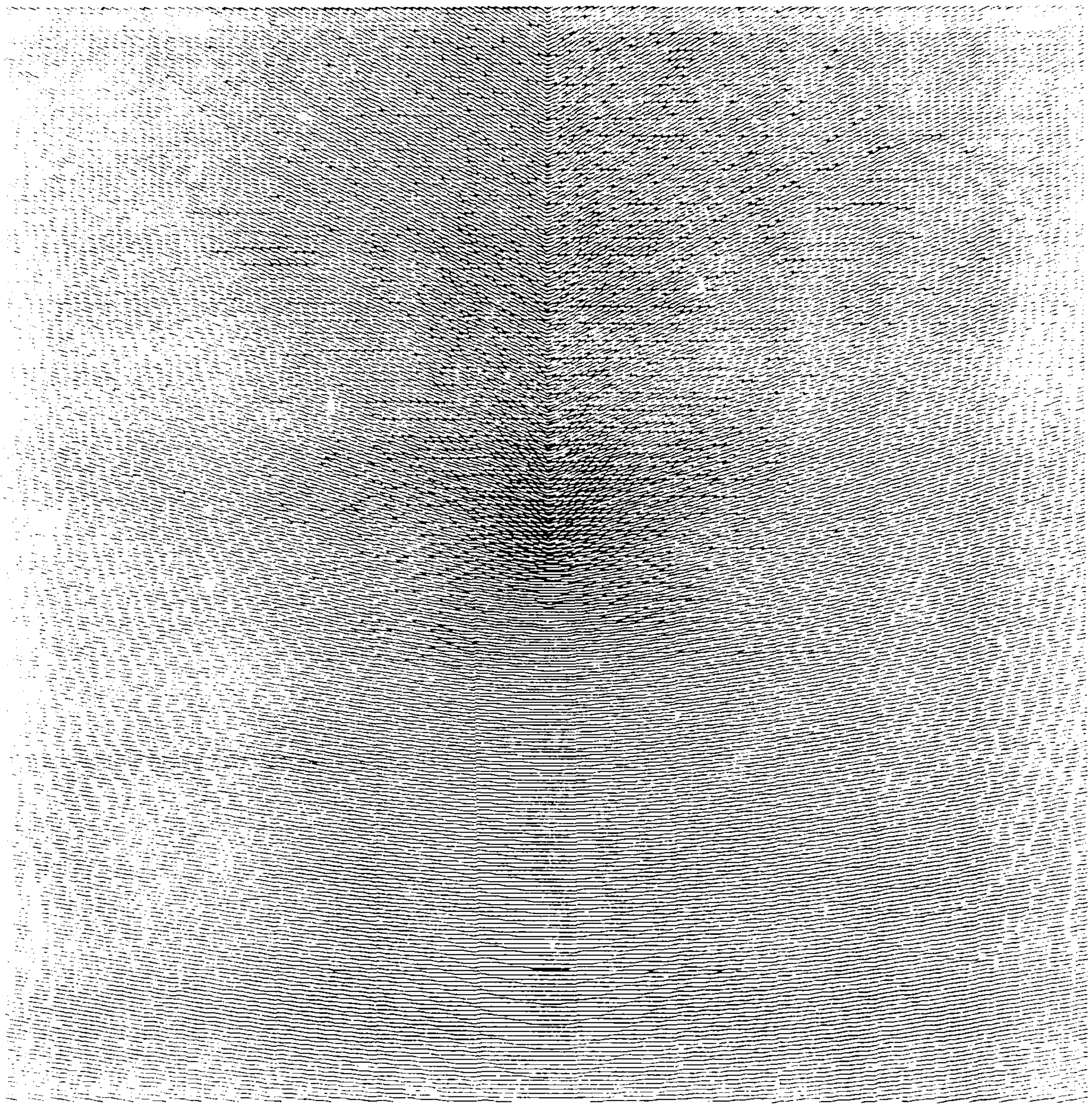


Figure B.1. Laser printer plot of the CGH. $\lambda = 0.6328\mu m$, $f_L = 300mm$. Photoreduce 20:1 to obtain an $\alpha$ of 35 lines/mm. Final dimensions: 10x10mm.

After the plot was obtained from the Imagen Laser Printer, a negative of the plot was produced at the Base Photo Lab. The negative was taken to the Cooperative Electronics Materials Processes Lab (room 1065) in building 125 to be reduced onto a high resolution film plate.

The Dekacon optical system was used to reduce the CGH and expose the film plate. A 3-inch Wray lens was placed into the system, as shown in Figure B.2, for a 20x reduction. A 20x reduction was obtained by setting the front box at 6.883 and the rear box at 27.000. The resulting image after these settings was 10x10mm with a carrier frequency (α) of 35 lines/mm.

The negative of the CGH plot was placed on the object screen and back illuminated with green light. Red Rubylith™

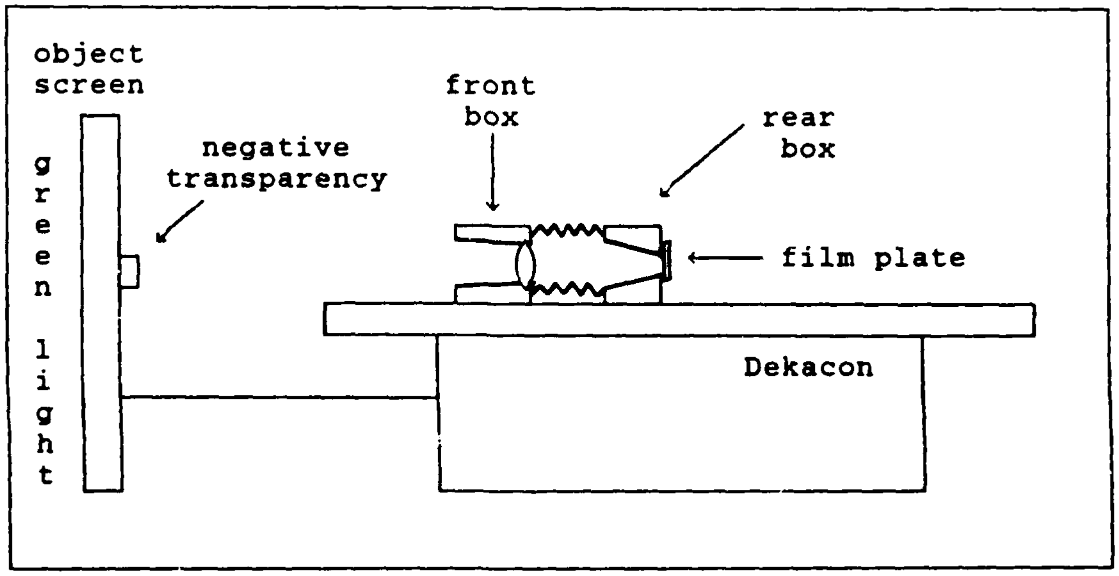


Figure B.2. Optical reduction of CGH.

material was placed around the plot to block all remaining light; therefore, the only light from the object screen was that which passed through the CGH plot.

Scratches were placed on a clear plate and placed in the image plane of the system. A microscope, placed behind the image plane, was used to focus on the scratches residing in the image plane while the front box was moved until the CGH plot came into focus with the scratches. When the plot and scratches were both in focus, the system was focused for the desired reduction. Note: it is very important that the scratched surface of the glass faces the object screen since the emulsion on the film plate will also face that direction.

Each Kodak 2x2 in. High Resolution Plate was exposed for 60sec by removing the shutter from the plate holder. The exposed plates were then developed for 40-60sec, depending on the type, age, and temperature of the developer. Best results were obtained by pulling the plate out of the developer every few seconds (after 30sec) and viewing it under red light until the grating pattern seemed "dark enough". The developed plate was then placed in the stop bath for 30sec followed by one minute in the fixer. A five minute rinsing in deionized water completed the development process. The developed plate was then viewed under a microscope to assure proper results (lines should be very dark and spacing between lines should be very clear).

## Bibliography

1. "AT&T Truevision Advanced Raster Graphics Adapter; Targa 8; Software Tools Notebook," operation manual for the frame grabber system: AT&T Electronic Photography and Imaging Center, Release 3.0 (November 1986).

2. Casasent, David. "Coherent optical pattern recognition: a review," Optical Engineering, 24: 26-32 (January 1985).

3. Casasent, David and A. Furman. "Sources of Correlation Degradation," Applied Optics, 16: 1652 (1977).

4. Casasent, David and Hironob Okuyama. "A High-Dimensionality Pattern Recognition Feature Space," SPIE Intelligent Robots and Computer Vision, 579: 245-256 (1985).

5. Casasent, David and D. Psaltis. "Deformation Invariant, Space-Variant Optical Pattern Recognition," Progress in Optics, 16: 289 (1978).

6. Casasent, David and others. "Real-time deformation invariant optical pattern recognition using coordinate transformation," Applied Optics, 26: 938-942 (March 1987).

7. Cederquist, J. and A. Tai. "Computer Generated Holograms for Geometric Transformations," Applied Optics, 23: (18) 3099-3104 (September 1984).

8. Flannary, David and others. "Real-time coherent correlator using binary magneto-optic spatial light modulators at input and Fourier planes," Applied Optics, 25: (4) 466 (February 1986).

9. "Functional Description for the Low Current Driver Card Set for the 128x128 Display System for the L135 Program," operation manual for the Litton Data Systems 128x128 LIGHT MOD.

10. Goodman, J.W. Introduction to Fourier Optics. New York: McGraw-Hill Book Company, 1986.

11. Gregory, Don A. "Real-time pattern recognition using a modified liquid crystal television in a coherent optical correlator," Applied Optics, 25: (4) 467-469 (February 1986).

12. Hughes, Kenneth D. Suitability and Applications of Liquid Crystal Televisions In Optical Pre-processors. MS thesis, AFIT/GE/ENG/86D-6. School of Engineering, Air Force Institute of Technology (AU), Wright Patterson AFB OH, December 1986.

13. Jensen, A.S. and others. "Transformation of image positions, rotations, and sizes into shift parameters," Applied Optics, 26: (9) 1775-1781 (1 May 1987).

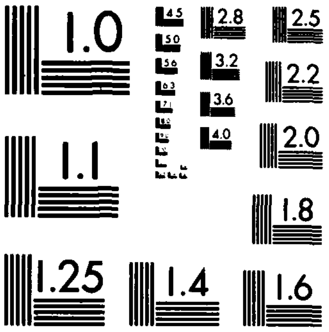
14. Kobel, W.G. and T. Martin. A Distortion-Invariant Pattern Recognition Algorithm. MS thesis, AFIT/GE/ENG/86D-20. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1986.

15. Neidig, David L. Optical Pattern Recognition Using Synthetic Discriminant Functions, MS thesis, GE/EE/86D. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1986.

16. Psaltis, Demetri and others. "Optical image correlation with a binary spatial light modulator," Optical Engineering, 23: 698-704 (December 1986).

17. Ross, W.E. and others. "Fundamental characteristics of the Litton iron garnet magneto-optic spatial light modulator," SPIE Advances in Optical Information Processing, 388: 55-64 (1983).

18. "Semetex Corporation SIGHT MOD Development System Operations Manual," for the IBM-PC/SIGHT MOD interface board.


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

## VITA

Michael W. Mayo was born on 28 August 1964 in Wiesbaden, Germany. He graduated from O'Fallon Township High School in O'Fallon Illinois in June 1982. Having earned an Air Force ROTC scholarship, he attended the University of Central Florida in Orlando, Florida, graduating in May 1986, with honor. He received the degree of Bachelor of Science in Engineering, electrical engineering option with a specialty in electro-optics. Upon graduation, he received a commission in the USAF through the ROTC program. After commissioning, he entered the Electro-optics Masters Program in the School of Engineering, Air Force Institute of Technology, in May 1986.

Permanent address: 205 Dartmouth Dr.
O'Fallon, Il 62269

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for public release; distribution unlimited. |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>AFIT/GEO/ENG/87D-1 | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>School of Engineering — 6b. OFFICE SYMBOL (If applicable) AFIT/ENG | 7a. NAME OF MONITORING ORGANIZATION |
| 6c. ADDRESS (City, State, and ZIP Code)<br>Air Force Institute of Technology<br>Wright-Patterson AFB, Ohio 45433 | 7b. ADDRESS (City, State, and ZIP Code) |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION — 8b. OFFICE SYMBOL (If applicable) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c. ADDRESS (City, State, and ZIP Code) | 10. SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO. |

11. TITLE (Include Security Classification)
COMPUTER GENERATED HOLOGRAM AND MAGNETO-OPTIC SPATIAL LIGHT MODULATOR FOR OPTICAL PATTERN RECOGNITION (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)
Michael W. Mayo, B.S., Second Lieutenant, USAF

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Year, Month, Day) | 15. PAGE COUNT |
|---|---|---|---|
| MS Thesis | FROM ______ TO ______ | 1987 December | 96 |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | Optical Pattern Recognition, Computer Generated Hologram, Spatial Light Modulator |
| 09 | 05 | | |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: COMPUTER GENERATED HOLOGRAM AND MAGNETO-OPTIC SPATIAL LIGHT MODULATOR FOR OPTICAL PATTERN RECOGNITION

Thesis Chairman: Dr. Matthew Kabrisky
Professor of Electrical Engineering

[illegible] release: IAW AFR 190-1.
[illegible] WOLAVER
31 Dec 87
Dean for Research and Professional Development
Air Force Institute of Technology (AU)
Wright-Patterson AFB OH 45433

| | |
|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT<br>[✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS | 21. ABSTRACT SECURITY CLASSIFICATION<br>UNCLASSIFIED |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>Dr. Matthew Kabrisky | 22b. TELEPHONE (Include Area Code) 513-255-3576 — 22c. OFFICE SYMBOL AFIT/ENG |

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

This thesis investigates the integration of an optical system for real-time position, scale, and rotation invariant pattern recognition. Specifically a Litton Magneto-Optic Spatial Light Modulator is interfaced to a Zenith 248 microcomputer and AT&T frame grabber. A user interface written in C allowed arbitrary patterns to be written to the SLM by a user. Patterns can be generated in computer code, transferred from a CCD camera (via the frame grabber), taken from an image file on computer disk, or taken from a VAX image file. The ability to locally generate a computer generated hologram using the computer generated interferogram method on the Sun workstations and Imagin laser printer was developed. The CGH developed was used in an optical coordinate transformation to create a position, scale, and rotation invariant feature space. CCD cameras were used throughout the optical processor for display/analysis. The entire system was developed and tested using various templates and real scenes.

END
DATE
3-88
DTIC